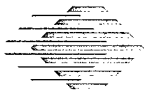
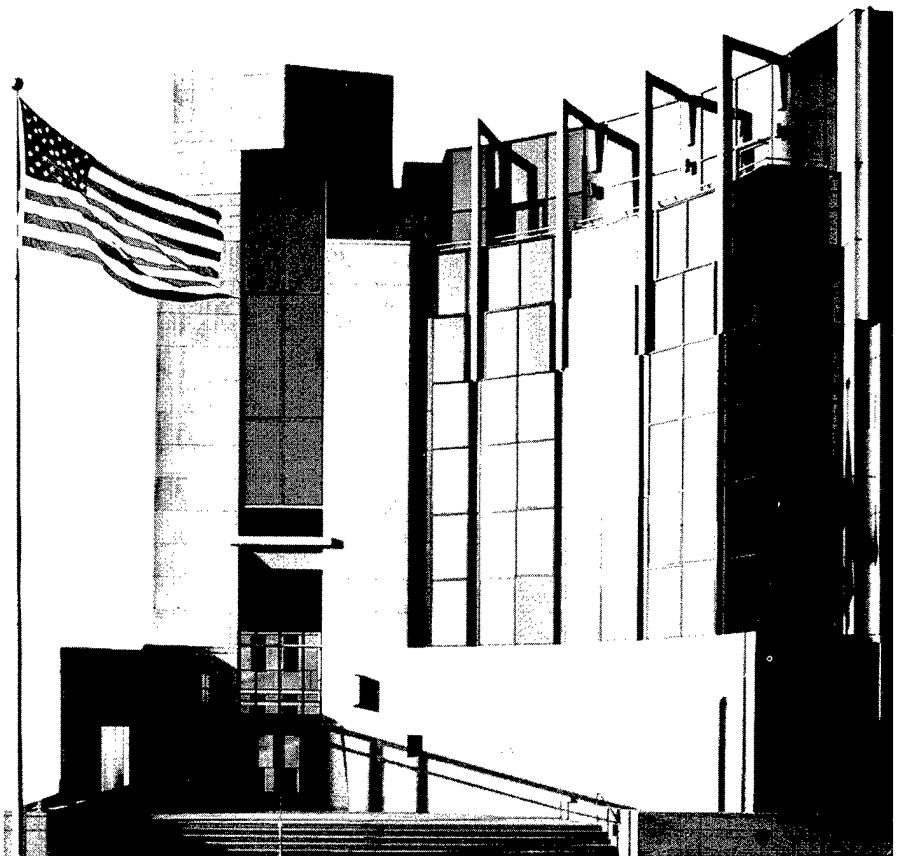


Carnegie Mellon
Software Engineering Institute

SEI Independent Research and Development Projects

Steve Cross
Eileen Forrester
Scott Hissam
Rick Kazman
Linda Levine
Rick Linger
Tom Longstaff
Ira Monarch
Dennis Smith
Kurt Wallnau

October 2002



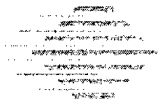
TECHNICAL REPORT
CMU/SEI-2002-TR-023
ESC-TR-2002-023

Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment, or administration of its programs or activities on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state, or local laws or executive orders. However, in the judgment of the Carnegie Mellon Human Relations Commission, the Department of Defense policy of "Don't ask, don't tell, don't pursue" excludes openly gay, lesbian and bisexual students from receiving ROTC scholarships or serving in the military. Nevertheless, all ROTC classes at Carnegie Mellon University are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Obtain general information about Carnegie Mellon University by calling (412) 268-2000.



Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

SEI Independent Research and Development Projects

CMU/SEI-2002-TR-023
ESC-TR-2002-023

Steve Cross
Eileen Forrester
Scott Hissam
Rick Kazman
Linda Levine
Rick Linger
Tom Longstaff
Ira Monarch
Dennis Smith
Kurt Wallnau

October 2002

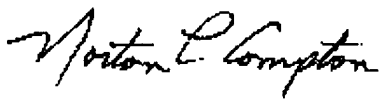
SEI Director's Office

This report was prepared for the

SEI Joint Program Office
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER



Norton L. Compton, Lt Col, USAF
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number F19628-00-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

List of Figures

List of Tables

Abstract

Each year, the Software Engineering Institute (SEI) undertakes several Independent Research and Development (IR&D) projects. These projects serve to (*a*) support feasibility studies investigating whether further work by the SEI would be of potential benefit and (*b*)support further exploratory work to determine if there is sufficient value in eventually funding the feasibility study work as an SEI initiative. Projects are chosen based on their potential to mature and/or transition software engineering practices, develop information that will help in deciding whether further work is worth funding, and set new directions for SEI work. This report describes the IR&D projects that were conducted during fiscal year 2002 (October 2001 through September 2002).

1 Introduction

1.1 Purpose of the SEI Independent Research and Development Program

Software Engineering Institute (SEI) Independent Research and Development (IR&D) funds are used in two ways: (1) to support feasibility studies investigating whether further work by the SEI would be of potential benefit and (2) to support further exploratory work to determine if there is sufficient value in eventually funding the feasibility study work as an SEI initiative. It is anticipated that each year there will be three or four feasibility studies and that one or two of these studies will be further funded to lay the basis for the work possibly becoming an initiative.

Feasibility studies are evaluated against the following criteria:

- Mission criticality: To what extent is there a potentially dramatic increase in maturing and/ or transitioning software engineering practices if work on the proposed topic yields positive results? What will the impact be on the DoD?
- Sufficiency of study results: To what extent will information developed by the study help in deciding whether further work is worth funding?
- New directions: To what extent does the work set new directions as contrasted with building on current work? (Ideally, we want a mix of studies that build on current work and studies that set new directions.)

At a DoD meeting in November 2001, the SEI's DoD sponsor approved a set of thrust areas and challenge problems to provide long-range guidance for the SEI R&D program, including its IR&D program. The thrust areas are survivability/security, interoperability, sustainability, software R&D, metrics for acquisition, acquisition management, and commercial off-the-shelf products. Though IR&D projects for FY2002 were selected prior to this guidance, the new studies initiated in FY2002 are focused on three of the thrust areas. The call for IR&D proposals in FY2003 will be based on these thrust areas and challenge problems.

1.2 Overview of IR&D Projects

The following feasibility studies were initiated in late FY2001 under the IR&D program:

Agent-Based Architectures: Intelligent agents are critical to making global and local resource allocation decisions that optimize the utility provided by resource-consuming services. Such agents can support the cooperative control of multiple applications or services with a uniform approach to scheduling multiple types of resources, including memory, power, processing, communication, and I/O devices. This study examined the application of adaptive agent-based architectures for mobile devices.

Analysis of Enterprise Integration Applications: Enterprise integration has the goal of providing timely and accurate exchange of consistent information between business functions to support strategic and tactical business goals in a seamless manner. As the DoD and all organizations take an increasingly global perspective, the need for integration between systems, many of which were originally developed as stove-pipe solutions for individual organizations, has become more urgent. This IR&D project has been investigating whether breakthroughs are possible in addressing the problem of the integration of information systems across an enterprise.

Flow-Service-Quality Systems Engineering: Foundations for Network System Analysis and Design: The purpose of this IR&D study was to investigate a unifying approach to large-scale network system analysis, specification, design, verification, implementation, and operation. The result of the research is an emerging technology called Flow-Service-Quality (FSQ) engineering. It focuses on complexity reduction and survivability improvement, and provides engineering and management foundations for network system development.

Several feasibility studies were started late in FY2000 and continued into FY2002:

Fusion: This project was designed to study the feasibility of multi-source data fusion for the predictive analysis of network intrusions. The goal was to identify and exploit data sources to gain insight into the likely targets and behaviors involved in network intrusions, particularly in *non-traditional networks*, which are networks that are temporary, ad-hoc, or special purpose. By gaining this insight, a number of defensive and preventative strategies become possible, with a consequent lessening of the large amount of resources presently dedicated to network defense and recovery from intrusions.

Open Source Software: Open source software (OSS) is emerging as the software community's next "silver bullet." The goal of this study was to examine OSS development and management practices to determine their viability and applicability to DoD systems.

Predictable Assembly from Certifiable Components: The purpose of this study was to address the fundamental challenge of building systems from components: predicting the properties of systems from their parts, strengthening the correlation between component properties

and system predictions based on them, and establishing a framework for gradual and sustained improvement in certification and prediction.

Quality Software Development @ Internet Speed: The purpose of this study was to examine how Internet software is developed today, with particular emphasis on the development processes used by Internet companies and the extent to which these processes are different from traditional software development processes.

Learning from Software Development and Acquisition Failures: This project was initiated to investigate how the SEI could make a strategic difference in preventing or reducing software-intensive system failure. Participants evaluated how methods for identifying, collecting, evaluating, analyzing, and synthesizing information about software failure can be used to detect patterns and trends related to software failures.

Technology Change Management in High-Maturity Organizations: The goal of this project was to determine which practices used by high maturity organizations to introduce software engineering innovations are most effective and how they might be captured for wider dissemination to other organizations. Improving the ability of organizations to adopt new technology is potentially one way the SEI could accelerate the transition of technology into widespread use.

These projects are described in detail below in the same order as they are listed here.

2 Agent-Based Architectures

2.1 Purpose

In our research, we are examining the application of adaptive agent-based architectures for mobile devices. The purpose of the adaptation is to make these devices more usable for the end user, despite the user's changing needs and the changing state of the device and its environment. To achieve adaptation, we are investigating the use of utility—subjective value to the end user—and a utility-based approach to resource allocation. By choosing "utility" as the central concept upon which adaptation rests (rather than, say, more traditional notions such as maximization of resource utilization or throughput), we can create systems that adapt in ways that more closely mirror the user's constantly changing needs and the constantly changing environment in which mobile devices operate. For example, mobile devices must endure widely varying network connectivity and quality of service conditions, and they must be constantly frugal about the power that they consume from their batteries.

Clearly the design space in creating such systems is enormous: the designer may want to experiment with different architectural configurations of the hardware and software components, and there are a large number of quality attributes that the user may want to have optimized (communication speed, battery life, central processing unit [CPU] performance, etc.). To aid in our research, we have built a testbed simulator, which simulates the various architectural components of a mobile device (both hardware and software), the user, and the environment in which the mobile system is operating.

We created the testbed simulator for the following purposes:

- to provide a tool through which one can explore methods for incorporating user preferences on mobile devices to maximize the utility of services provided
- to explore different scheduling and resource allocation strategies
- to do performance modeling
- to systematically explore the impacts of architectural alternatives on resource utilization and the maximization of utility

We are already seeing the commercial presence of cell phones and personal digital assistants that combine voice and data communication and other applications (such as video) on a single device. This trend appears to be steadily increasing. But the methods of building these

systems have not evolved significantly and do not differ greatly from the methods for building non-mobile embedded systems. For example, in traditional scheduling and resource allocation approaches, the relative priorities of computational elements (such as threads, processes, and programs) are the information that is used to make scheduling decisions. However, on a personal communication device, this approach needs to be extended towards a *utility*-based resource allocation mechanism, where utility is purely user defined. In this view of the world, application services that have a high utility to the user receive a preferential allocation of resources. This utility can change according to the user's needs, the external environment, or the state of the system itself (such as the system's battery level).

A personal communication device serves multiple purposes. Each of these multiple purposes implies a change in a user's preferences. The user's mobility implies inevitable changes in the environment, affecting communication bandwidth and perhaps even sources of energy and other devices with which one could collaborate. Mobility provides the opportunity for dynamic coalitions of computational resources as mobile devices come into contact with each other. Mobility also leads to changes in the dominant use of the device. A utility-based resource allocation must be able to dynamically adapt to these changes.

2.2 Background

Typically, the runtime resources for exiting mobile systems are highly constrained; examples of such resources include CPU memory, bandwidth, screen real estate, and battery. In the current process for building mobile systems, resource allocation decisions are almost always fixed at the time of system creation as a means of managing their constrained resources with relatively low overhead. However, this situation is arguably changing, both for the better and for the worse. For the better, modern mobile systems are becoming more powerful, in terms of the computation and communication resources upon which they can call. Taking advantage of this power, we can begin to consider designing systems that adapt to their environments at runtime, rather than simply having their resource allocation decisions fixed forever. For the worse, at the same time as mobile systems' resources are increasing, the demands being placed upon mobile devices are also increasing dramatically. It is our contention that this will *always* be the case: despite increases in their computational and communication abilities, mobile devices will always be underpowered with respect to changing and ever-increasing users' needs. For this reason, such systems need effective methods to manage and control their resources at runtime, particularly in the face of changing environmental conditions and user needs.

2.3 Approach

Mobile applications are expected to be able to run at different fidelity levels providing a different level of utility to the user and having different resource requirements [Noble 97]. For

example, if a user is browsing the Web, different fidelity levels might take the form of text-only display, text with minimal graphics, and text with full graphics. Voice communications might have different levels of fidelity corresponding to different CODECs (COmpression, DECompression) that run at different data rates. Video communications might have different resolutions, color depths, and numbers of frames per second. In each case, the resource requirements for these applications are different, and the resulting utility that they provide to the user may be different. In addition, a user's subjective view of the utility of these different levels of fidelity might change depending on the situation. For example, users might demand high voice quality for an important business call with a client, but they might be perfectly content with a low voice quality when checking sports scores or when receiving unsolicited calls. Even within a single run of a single application, the fidelity demanded by a user might change dynamically.

Finally, the user's environment is changing dynamically. The battery power on the system is always changing, and a user might add or remove a battery. The communication signal strength required for sending or receiving data might vary both over time and with changes in the user's location. Available network bandwidth might also vary considerably over time, due to normal fluctuations in the load on the network. To be able to conduct research in determining appropriate software structures for dealing with all of these complexities, we are exploring different architectural alternatives for such mobile devices; to do this easily and inexpensively, we need a simulation testbed.

The research questions that we propose to answer from this simulation testbed can be categorized into two broad categories:

- **Usability**: Is it possible to build a system that incorporates user preferences and dynamically optimizes the assignment of resources to maximize user satisfaction? What are the constraints on building this type of system? What are the resource consumption characteristics of various processes (including the optimization process) and how do they interact? How can user preferences be accurately captured and transformed into information that is appropriate for making scheduling decisions?
- **Architecture**: What are the various architectural structures that can be used to provide dynamic allocation of resources based on user preference? Is it possible to prescribe architectural designs for different types of mobile devices, based on the desired characteristics of those systems? How do centralized and decentralized architectures differ and how are they related?

The problem of creating such architectures is challenging because many aspects of the architecture are stochastic. For example, the environment is stochastic, and the user is stochastic with respect to the changing environment. For this reason the mobile device must constantly adapt to the changing environment, to its own changing state (for example, a low battery condition), and to the needs of the user. Therefore, it is crucial for the testbed to be able to easily support the specification and simulation of different architectural choices so that these

choices can be compared in a disciplined and repeatable way; by using a simulator, these choices can be explored before building the system itself.

2.4 Views of the System

Typically, describing any software architecture requires multiple views of a system: for example, the functional, runtime, and implementation views [Bass 98].

2.4.1 Functional View of the System

From a purely functional perspective, the testbed consists of three layers: the resource layer, the task layer, and the environment layer.

- **resource layer**: This lowest layer simulates hardware resources. This layer consists of the basic resources that are needed to run an application on a mobile device.
- **task layer**: This layer consists of various modules that drive the mobile device to perform useful services on behalf of the user. Tasks consume resources and provide utility to the user. An essential function within this layer is the scheduling module that prescribes the resource allocation across applications over a period of time, and how allocations will change according to changes in the environment or in user-indicated utility.
- **environment layer**: This layer simulates the environment within which the mobile device operates. This layer simulates a stochastic environment from the perspective of the mobile device, including user events, changes in network bandwidth and availability, and external application events (such as the arrival of a phone call or a short message).

2.4.2 Runtime View of the System

In this section we describe in more detail how the various modules within the layers of the architecture are scheduled and interact to accomplish the goals of the simulation. The runtime view of the system is described in Figure 1.

The simulation controller is the runtime heart of the system. Currently it reads a set of simulation script files that determine the nature of the resources and tasks, their configuration, the user activities, and the environmental events that the system is to simulate. The script files determine the architecture of the system being simulated. By changing these files, we can easily simulate many different architectures with different task and resource characteristics and different scheduling disciplines. This configurability makes the testbed an ideal vehicle for experimenting with architectural alternatives at a very low cost.

The simulation controller causes application modeling to execute the user, the environment, and the various simulated tasks. These tasks in turn make use of simulated resources, via the auspices of some scheduling mechanism. All communication among the simulated compo-

nents takes place through the publish/subscribe service. This makes it simple to change the architecture of the simulated system, because no component directly depends on, or even knows of the existence of, any other component.

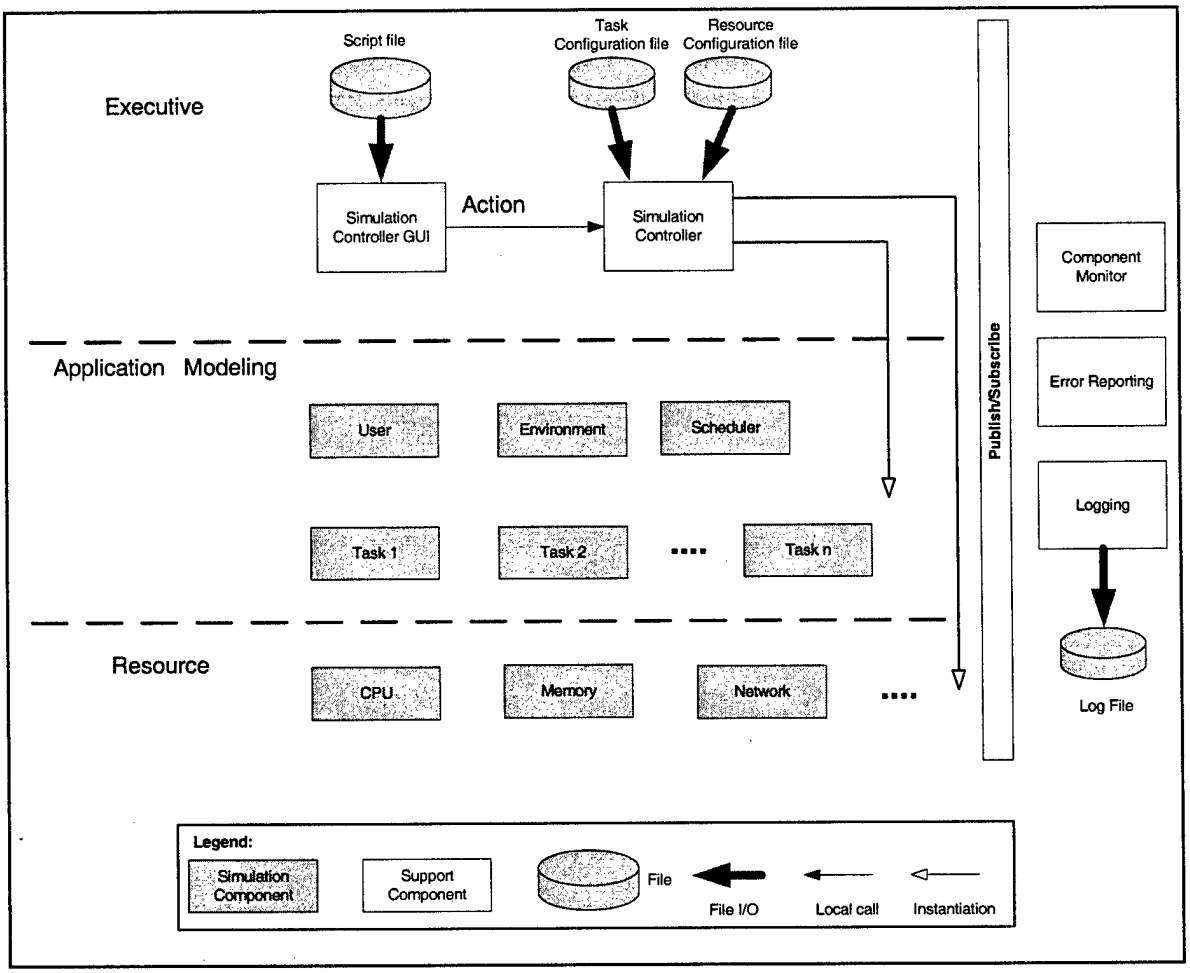


Figure 1: Runtime View of the Simulation Testbed

2.4.3 Implementation View of the System

The simulator consists of three key component types—resource, task, and scheduler—each of which is represented by a distinct Java™ class. The resource component type represents the usable system resources such as battery, CPU, memory, and network bandwidth, of which the battery is a special kind of resource (draining over a period of time.) The resources are targets for competition between task components. The task component type simulates each unit of meaningful work such as a voice call, file download, or playing of music. The scheduler component lies between these two groups and arbitrates the competition between tasks trying to get their required resources.

™ Java is a trademark of Sun Microsystems, Inc.

2.5 Collaborations

Rick Kazman is the member of the SEI technical staff who has worked on this project. Jayatirtha Asundi, an SEI visiting scientist, also contributed. The main external collaborator on the project has been Alexander Ran, Principal Research Scientist at Nokia Corporation. He is providing his own support for the project. A small amount of collaboration has also take place with the Aura project, headed by David Garlan in SCS, and with the QRAM project, headed by Peter Feiler at the SEI.

2.6 Evaluation Criteria

The evaluation criteria that we set out for this project in our original proposal were as follows:

- At least two organizations (one commercial and one DoD) build directly upon this work in continued experimentation and research. Likely candidates are
  - Nokia: We have been working with them since September 2000.
  - Motorola: We have been in contact with Landis/Vesudaven since January 2001.
  - Defense Advanced Research Projects Agency (DARPA) Agent-Based Systems Program

This criterion has not yet been met, but the project is not yet completed. We are still working on publicizing the research and actively collaborating with our industry partners to find applications of the testbed.

- At least one paper is accepted in a major conference or journal.

As of this time, we have had one paper accepted at a major conference (the 2002 International Conference on Software Engineering Research and Practice [SERP '02]) and one paper accepted in a journal (*Computer Standards and Interfaces*).

- As least one conference session, track, or workshop is devoted to this issue and motivated by SEI work.

We will be proposing a mini-track at the Hawaii International Conference on System Sciences (HICSS) on this topic for next year. We have every reason to expect that this will be accepted.

2.7 Results

This IR&D project is still in progress, but we already have some successes to report:

- Nokia is still involved with the work and is looking at ways of building on it.
- We have made one conference publication: Kazman, R.; Asundi, J.; and Ran, A. "Adaptable Architectures for Mobile Systems." *Proceedings of the International Conference on*

Software Engineering Research and Practice. Las Vegas, NV, June 2002. Las Vegas, NV: Computer Science, Research, Education and Applications, 2002.

- We have made one journal publication (an extended version of the conference paper), accepted for publication in *Elsevier's Computer Standards and Interfaces* journal.
- We have submitted another paper to the Hawaii International Conference on System Sciences (HICSS).

Currently we are experimenting with the implications of centralized versus decentralized scheduling disciplines, and on the implications of different models of distributed resources requesting and consuming resources. To create truly distributed resource management, we must experiment with different negotiation schemes where the tasks themselves are intelligent agents, capable of negotiating for resources on their own behalf. Being able to support this dramatically different architectural style is a significant test of the generality of the testbed.

At the moment, the simulation testbed runs from script files. While this is adequate for testing the proper working of the testbed itself, it does not provide sufficient support for broad test coverage of the simulated components and architectures because a script file will always run deterministically, producing the same result. To address this shortcoming, we are adding in a "simulated user" that is non-deterministic. A major research area, therefore, is to determine appropriate parameterizable models to represent the user. We are also adding an environment simulation component, to simulate changing network conditions, equipment failures, and so forth. Since this testbed is meant to obviate the need to build hardware devices for testing, we must be able to adequately simulate the hardware, its behavior, and the environment in which it runs.

To increase the usability and user satisfaction of mobile devices, we propose to use the utility judgments of the user. Clearly much of the success of a utility-based scheduling discipline will depend on how the utility levels are set; determining whether these truly reflect the desires of the user and capturing the utility judgments of the user are of prime importance. In our simulation testbed, we have created the provision for a "training" mode. In this mode, we expect to capture and infer the utility values for various resource consumption levels by observing the user at work. This will, we expect, result in far superior utility settings than doing this manually, which would certainly be an ad hoc process. For mobile devices, we expect the user to run the device's training mode when she buys the device—just like a user trains her voice recognition software to reduce errors in transcription.

2.8 Publications and Presentations

Kazman, R.; Asundi, J.; and Ran, A. "Adaptable Architectures for Mobile Systems." *Proceedings of the International Conference on Software Engineering Research and Practice.* Las Vegas, NV, June 2002. Las Vegas, NV: Computer Science, Research, Education and Applications, 2002.

Kazman, R.; Asundi, J.; Kim, J.; and Sethananda, B., "A Simulation Testbed for Mobile Adaptive Architectures," *Computer Standards and Interfaces Journal*, accepted for publication, 2003.

3 Analysis of Enterprise Integration Applications

3.1 Purpose

This IR&D project has been investigating whether breakthroughs are possible in addressing the problem of the integration of information systems across an enterprise. Enterprise integration has the goal of providing timely and accurate exchange of consistent information between business functions to support strategic and tactical business goals in a manner that appears to be seamless.

The need for integrated battlefield management, the integration of intelligence from different sources, and the integration of business functions (such as materiel, human resources, and supply chain information) has been recognized by the Department of Defense (DoD) as important for some time. As the DoD and all organizations take an increasingly global perspective, the need for integration between systems, many of which were originally developed as stove-pipe solutions for individual organizations, has become more urgent.

The criticality of enterprise integration has been cited by Joint Vision 2020 [JV 00, p.21]. In addition, every federal government organization has been mandated by the Clinger–Cohen Act of 1996 [Public Law 96] to appoint a Chief Information Officer (CIO) responsible for "developing, maintaining and facilitating the implementation of a sound and integrated information technology architecture."

The enterprise integration problem is equally significant within the corporate world. Business survival in today's climate often depends on the ability to integrate applications effectively across the value chain of the business. As computer-intensive applications have become more pervasive, the ability of government organizations and corporations to achieve their strategic goals and objectives hinges on their ability to access, analyze, and process data quickly, accurately, and effectively.

3.2 Background

The lack of enterprise integration costs billions of dollars per year. In many cases, legacy systems are hampered by inconsistencies and inefficiencies resulting in the fact that integration

is difficult because of incompatibilities across applications and because each application owns and controls its own data. Zachman reports data from a number of sources showing that between 20% and 40% of all labor costs in the United States are dedicated to the gathering, storage, and reconciliation of data, and that 70% of the lines of software code in corporate software systems are dedicated to moving data from system to system [Zachman 97].

To address the problem, a number of different approaches are in use, including developing enterprise architectures, using Web-based and middleware approaches, and inserting an Enterprise Resource Planning (ERP) solution. While each type of approach offers a potential step forward, no single approach offers a systematic solution. Some outstanding problems with each type of approach are outlined below.

Enterprise architectures are high-level definitions of the data, applications, and technology needed to support the business. The result of the process is a long-range plan to support the overall goal of enterprise integration. However, there are several problems with the current set of approaches. The linkage to the software architecture required to implement the plans is often left for future work, resulting in a disjunction between the planning stage and the implementation stage. In general there is not clear guidance on this step. As a result, initial plans are often developed which later flounder for lack of a clear connection to actual system development and for the lack of a detailed implementation plan. Few enterprise architecture plans have actually been implemented successfully.

A number of promising technology approaches are maturing rapidly. Finkelstein and Aiken have recently proposed the use of Web-based technologies in conjunction with middleware to assist with the problem of enterprise integration [Finkelstein 99]. The potential of the Web as a portal to corporate data provides a mechanism to allow for searching and retrieving inquiries and for accessing information about orders, order fulfillment, and customer data.

Early uses of the Web using Hypertext Markup Language (HTML) permitted static information to be displayed, but they did not provide benefit for enterprise integration. Later the introduction of Java provided for portability across hardware platforms and operating systems. Technologies such as Enterprise JavaBeans (EJB) simplified the development of middleware components that support transactions, security, and database connectivity.

While the use of Web and middleware technologies to support enterprise integration has strong potential, the application of these technologies is only beginning to be explored. There is not a good understanding of the appropriate role of middleware technologies that can enable pragmatic, loosely coupled integration. However, there are not clear guidelines on when specific types of technologies are appropriate, how to integrate a solution set within the technical and organizational structure most effectively, and how a set of choices will affect future technology options.

ERP solutions, such as SAP, PeopleSoft, and Oracle, are advertised as providing one-stop shopping that will integrate all of an organization's information technology (IT) needs. The problem of enterprise integration is sometimes assumed to be primarily that of selecting and fine-tuning a specific ERP solution. However, decisions on selecting an ERP solution are often made without a detailed technical understanding of exactly what the ERP provides, how it will support an organization's processes, and how it will integrate with an organization's existing legacy systems.

Addressing these issues is critical for progress in current efforts at enterprise integration. The dozens of current large-scale efforts at enterprise integration within the DoD and other organizations tend to struggle with some subset of these problems.

3.3 Approach

The study examined a number of the critical questions and issues that need to be addressed in order for enterprise integration to be a routine practice, and it identified potential ways to address the critical questions and issues. Some of the questions and issues include the following:

- What are the basic issues of enterprise integration?
- Which issues are currently being addressed adequately?
- Where are the gaps in our current understanding of the state of the practice?
- Where are the gaps between the state of the art and the state of the practice?
- Are there specific areas where significant progress can be made?

To address these issues, the study has analyzed the broad topic of enterprise integration. It has focused on getting an overall understanding of the field and determining whether potential new approaches might provide high leverage for practitioners to make progress. The final report of the group is the *Technical Roadmap for Enterprise Integration*, which will be available in October 2002. The Technical Roadmap will provide a summary of critical issues and will identify gaps in current research and practice, as well as potential areas where the SEI may be able to make a contribution. Some of the preliminary conclusions are summarized in Section 3.6.

As input for the Technical Roadmap, the following sets of activities have been undertaken:

3.3.1 Workshops

Centre for Advanced Study Conference (CASCON) Workshop

A workshop for the research and corporate community was held in conjunction with CASCON 2001 in Toronto in October 2001.Forty five people participated from IBM, Sun, the Natural Sciences and Engineering Research Council of Canada (NSERC), Bell Canada, University of Waterloo, University of Victoria, University of Alberta, and Queens University. IBM Toronto and Bell Canada have particularly indicated interest in future involvement.

The CASCON workshop focused on technology trends and identified outstanding issues that need to be addressed before substantial progress can be made.

The positive trends include

- net-centric computing
- Web as a portal to corporate data
- implications of electronic commerce
- Java-enabled portability across hardware platforms and operating systems
- maturity of Enterprise Application Integration (EAI) middleware (Websphere, Component Broker, Vitria, .NET) and platforms (such as Eclipse) for the effective integration of tools [Britton 01]
- potential of mark-up languages (such as XML) for linking structured and non-structured data [Finkelstein 99].

The outstanding problems include

- a need for an effective methodology for enterprise integration
- a lack of high-level awareness of issues
- a lack of generic solutions to the integration problem (integration solutions tend to be local)
- the extraction of interfaces from existing systems and identification of side effects
- a need for clear guidance for management decision making
- a need for an overall solution to distributed transaction processing on a Wide Area Network (WAN) (latency, wide-scale transaction, scalability)

DoD Workshop

The DoD workshop was held in May 2002 at the SEI Washington office. The workshop included 18 invited participants from the DoD and other government agencies. The workshop conclusions are folded into the discussion of results in Section 3.6.

3.3.2 White Papers

The team produced a series of informal white papers to raise an understanding of issues and provoke discussion. These white papers focused on the following topics:

- the role of net-centric computing in enterprise integration architectures
- customizable service integration in Web-enabled environments
- the role of enterprise portals in enterprise integration
- quantitative information technology portfolio management
- migrating and specifying services for Web integration
- annotated bibliographies

3.4 Collaborations

Dennis Smith, Liam O'Brien, Robert Seacord, Mario Barbacci, and Ed Morris are the members of the SEI technical staff who worked on this project. The feasibility study involved extensive contact with external collaborators. As mentioned earlier, the workshops involved participation with representatives from the research and corporate community, as well as the DoD community. Some of the collaborators who have expressed interest in further involvement include

Lt. Col Joe Besselman, Global Command and Control System (GCCS)
Dr Loring Bernhardt, MITRE
Dr. Anna Liu, Commonwealth Scientific and Industrial Research Organisation (CSIRO)
Dr. Joe Jarzombek, Office of the Secretary of Defense (OSD)
Mr. John Gay, Air Force – Chief Information Officer
Mr. William Taylor, Office of Housing and Urban Development (HUD)
Mr. Bradford Eyre, Coast Guard
Mr. John Wunder, Lockheed-Martin
Dr. Marin Litou, IBM-Toronto
Dr. Raymond Paul, Command, Control, Communications and Intelligence (C3I)

In addition the IR&D team was structured to include external team members. These team members included

- Dr. Peter Aiken of Virginia Commonwealth University. Aiken is considered a leader in this area, and has written widely on the topics of enterprise integration, legacy system migration, data modeling and more recently on Extensible Markup Language (XML) portals. Aiken has been involved with a number of organizations that have attempted enterprise integration, and he has an understanding of the major concepts in the field, current strengths and weaknesses, and the key players.

- Dr. Kostas Kontogiannis of Waterloo University. Kontogiannis is recognized for his work in reengineering, migration of legacy systems, and the application of Web technologies, such as XML. He has directed large-scale commercial projects in applications of Web-based technologies to legacy systems.
- Dr. Atul Bhatt. Bhatt has done extensive work in IT strategic planning, technology integration planning, and enterprise architecture management of large-scale information systems for such companies as Nortel, Caterpillar, General Electric (GE), International Business Machines (IBM), Ford, Sprint, Becton Dickinson, and National Capital Region (NCR).

3.5 Evaluation Criteria

Success criteria include the following:

- development of a preliminary roadmap for progress in enterprise integration that integrates insights from a broad set of perspectives, including insights from SEI work
- successfully conducting two workshops eliciting input on needs and issues from DoD stakeholders, researchers and the corporate community
- establishment of interest among a community of researchers and practitioners that has promise to go beyond the period of the study, regardless of further SEI involvement
- development of at least four position papers that will elucidate basic concepts on topics that are sometimes confusing to practitioners
- an analysis of the potential longer term SEI role (If the SEI has a longer term role to play, this criterion will include development of an integrated SEI strategy for addressing enterprise integration, including a consensus among the SEI initiatives on the decisions that are made.)

3.6 Results

At a very high level, the study has identified the following basic problems:

- Many organizations are unable to determine an appropriate scope, strategy, and plan for an effort. This has often led to scope escalation, an inability to define an effective project, and an inability to get measurable results.
- There is not a common understanding of the role of an enterprise architecture. It is often seen as a task to perform in order to check off a box. The focus on the low-level details of an enterprise architecture obscures the purpose of the task. The task is often viewed as being a part of implementation, while its primary purpose is actually planning.
- Top-level managers often confuse an enterprise architecture with a software architecture and conclude that a software architecture is not necessary. This problem has led to significant churning, particularly within the DoD. It has also led to a failure to account for the critical role of quality attributes.

- While CIOs have been appointed to comply with the Clinger-Cohen Act, they tend not to be integrated within the services and are viewed as simply an additional level of bureaucracy whose approval is necessary in order to proceed.
- While middleware and technology solutions can offer significant advantages, decisions are often made at the level of slogans and specific vendor solutions.
- The problem of enterprise integration is often assumed to be primarily that of selecting and fine-tuning a specific ERP solution. There is a gross underestimation of the task of integration with existing organizational processes and integration with legacy systems, migration of users, and deployment.

To address these issues, the problem of enterprise integration needs to be addressed at multiple levels. The unmet needs include

- a proven method for developing an effective scope for integration efforts. Currently, many efforts flounder because they fail to define an effective scope. The issue of scope is at the intersection of the unresolved problems described earlier, and addressing it offers one way to establish a closer connection between enterprise architectures and software architectures. As discussed earlier, while enterprise architectures are meant primarily as a tool for planning rather than implementation [CIO 01], in practice such efforts go to low levels of detail, take significant amounts of effort, and have little linkage to follow-on projects.
- decision rules for making choices on the types of technology that are most appropriate for specific types of efforts, including choices on ERP solutions, Web technologies, and middleware. The types of decisions where support is needed include choices on the type of technology that is appropriate, its relationship to existing systems, its impact on the processes of the organization, and the amount of effort that is required to adapt the technology. Although many technology solutions are available, there are not easily accessible guidelines for when to use different types of solutions.
- a clear understanding of how to migrate legacy systems successfully. Most enterprise integration efforts require migration from legacy systems. Currently, there is not a systematic understanding of how to migrate successfully from legacy systems. As a result, many organizations seriously underestimate the magnitude of the effort required.
- a comprehensive handbook on critical issues and choices
- a visible community that can offer role models, guidance, and transition support

These outstanding issues and unmet needs are addressed in more detail in the Technical Roadmap. The roadmap also identifies potential areas where future SEI effort in the area can result in the greatest impact.

3.7 Publications and Presentations

Smith, Dennis. *Technical Roadmap for Enterprise Integration* (CMU/SEI-2002-TN-028). To be published in 2002.

4 Flow-Service-Quality Systems Engineering: Foundations for Network System Analysis and Design

4.1 Purpose

Government, defense, and business enterprises are irreversibly dependent on large-scale network systems whose complexity frequently exceeds current engineering capabilities for intellectual control. The result has been persistent difficulties in system development, management, and evolution, as well as failures, intrusions, and compromises in operation. (The term "system" is used in the broadest sense to encompass hardware, software, and people.) These systems are characterized by very large-scale heterogeneous networks with often-unknown boundaries and components. Dynamic interconnectivity of systems-of-systems limits visibility and control of security and survivability. User task flows can traverse systems and boundaries with varying security and survivability characteristics. In addition, these systems must deal with uncertain commercial off-the-shelf (COTS) component function and quality, unforeseen behaviors and vulnerabilities, and unanticipated inter-system cascade failures. Complexity is compounded by the extensive asynchronous behavior of the virtually unknowable interleaving of communications among system components. It is often the case that the components of network-centric systems are complex systems in their own right, and must be dynamically integrated to provide coherent mission capabilities.

The purpose of the Flow-Service-Quality Systems Engineering IR&D study was to investigate a unifying approach to large-scale network system analysis, specification, design, verification, implementation, and operation. The result of this research is an emerging technology called Flow-Service-Quality (FSQ) engineering. It focuses on complexity reduction and survivability improvement, and provides engineering and management foundations for network system development. FSQ engineering comprises three related technologies: Flow Structures, Computational Quality Attributes, and Flow Management Architectures.

4.2 Background

The burden of unmastered complexity creates difficulties in understanding systems we have and in defining systems we need. It leads to loss of intellectual control when human capabilities for reasoning and analysis are exceeded. Intellectual control means understanding system behavior in all circumstances of use. It means orderly development and evolution, and no surprises in operation. Intellectual control does not mean the absence of uncertainty, failures, or compromises—they are inevitable—but the capability to deal with them through engineering technology. And it does not require slow and burdensome methods for development. On the contrary, intellectual control enables rapid development with confidence. Today, we face complexity and survivability issues on a whole new level. How systems fail has become as important as how they succeed. Complexity and survivability are closely related. Complexity diminishes survivability by masking potential failures and vulnerabilities and by hiding unforeseen access paths for intrusion. Survivability requires knowing system component dependencies, and defining system actions for component compromises and failures in all circumstances of use [Mead 00]. In short, survivability requires intellectual control.

More effective systems engineering methods are required across the life cycle for fast, precise, and predictable system development and evolution. A promising path lies in the definition of unifying mathematical foundations upon which to base engineering practices and automation support. These foundations must accommodate network system realities: highly distributed heterogeneous components; shifting boundaries and users; uncertain COTS component function and performance; unpredictable failures, disruptions, and compromises; and lack of information and control of component and network behavior. They must also accommodate enterprise needs for rapid development and evolution, predictable composition of all-scale components, and interoperability of systems to achieve mission objectives.

4.3 Approach

The primary focus of the FSQ IR&D study was the definition of mathematical semantics for engineering methods and representations that explicitly address the realities of network-centric systems that have challenged the best engineering efforts:

Network System Reality	Requirements for FSQ Semantic Foundations
Enterprise- and user-centric view of system services and operations	Engineering methods that begin with enterprise requirements and refine into designs and implementations
Multiple levels of abstraction in specification and design	Engineering methods for precise, scalable refinement, abstraction, and verification at all levels
Incomplete knowledge of behavior of network services	Engineering methods that are based solely on service responses, not complete service behavior
Uncertainties of COTS component function and quality attributes	Engineering methods to explicitly design for Uncertainty (Survivability) Factors of failure and compromise
Complexities of pervasive asynchronous behavior	Engineering methods and representations that are deterministic for human understanding and analysis
Constantly changing values of quality attributes in operation	Computational approach to quality attributes to accommodate dynamic attribute behavior in execution

These foundations can have a major impact on network system acquisition, development, operation, and evolution, all of which are key areas of the DoD challenge problems.

4.4 Collaborations

The FSQ IR&D study brought highly capable researchers with proven track records into collaboration with the SEI. The Survivable Systems Working Group (SSWG) was established to conduct the study, with participation by the University of Central Florida and the University of South Florida. In addition, a related project has been initiated with a DoD organization. A focus area of this project is application of FSQ concepts to survivability analysis of mission-critical essential services in existing systems. The methods defined involve extraction of flows from system architectures to reveal dependencies and potential points of failure and compromise. These findings focus and prescribe survivability improvements to system architectures.[1]

4.5 Evaluation Criteria

The objective of the FSQ IR&D study was to develop semantic foundations for engineering methods and representations that address the network system realities described above. That objective has been achieved, as discussed briefly below and in more detail in the publications listed in Section 4.7. The next step in FSQ development is to create engineering examples,

[1] Linger, R. *Essential Service and Sense-and-Respond Control Models* (CMU/SEI-2002-SR-007). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002 (restricted distribution).

language structures, and engineering practices as artifacts for technology transfer and application.

4.6 Results

The study focused on developing three engineering concepts that address the challenging realities of network-centric systems. Taken together, these concepts comprise an engineering discipline for large-scale systems:

- Flow Structures: User task flows and their refinements into system service uses can provide unifying engineering foundations for analysis, specification, design, verification, and implementation of functionality and quality attributes.
- Computational Quality Attributes: Quality attributes, such as reliability, availability, and survivability, can be associated with both flows and the system services they invoke, and can be specified as dynamic functional properties to be computed, rather than as static, a priori predictions of uncertain utility in the fast-changing environment of system execution.
- Flow Management Architectures: Flow Structures and Computational Quality Attributes support canonical architecture frameworks that manage flows, network services, and their quality attributes in execution.

Details of Computational Quality Attributes and Flow Management Architectures can be found in the published papers cited in section 4.7. The following discussion focuses on Flow Structures.

Flow Structures are compositions of system services that carry out user tasks to accomplish enterprise missions. They employ mathematical semantics that were defined in this study to preserve important deterministic properties for precise human understanding and analysis, despite the underlying uncertainty and asynchronism of network behavior. Flow Structure engineering requires designing for unpredictable events that can impact mission survivability. And Flow Structures provide a vehicle for specification and management of quality attributes. Thus, the first-class concepts of flow, service, and quality are the essential and primary artifacts of the emerging discipline of Flow-Service-Quality (FSQ) engineering [Hevner 02a, Hevner 02b, Linger 02a].

4.6.1 Flow Structures

Distributed information systems are usefully viewed as networks of asynchronously communicating components that provide services whose functions can be combined in various patterns to satisfy enterprise mission requirements. System services include all the functional capabilities of a network system, from protocols, operating systems, and middleware to databases and applications. The sequencing of system services in user task flows can be mapped into compositions of network hardware, software, and human components that provide the

services. These compositions are end-to-end traces that define slices of network architectures whose net effect is to carry out operations that satisfy user requirements [Hevner 01]. The top of Figure 2 depicts refinement of user task flows into uses of system architecture components. Flows are essentially procedures that define compositions of service uses at levels of abstraction ranging from an enterprise mission to system implementation. Flows can specify integration and traversal of multiple systems and components, as shown in the bottom of Figure 2. Flows can be expressed in a simple set of control structures, including sequence, alternation, iteration, and concurrent structures, and can be refined, abstracted, and verified with precision [Prowell 99]. Flows invoke services, which can be refined into flows, etc., in a recursive process that employs identical methods at all levels of specification and design.

Flow Structures are based on a function-theoretic semantic model. Flows and their constituent control structures are regarded as rules for mathematical functions (or relations), that is, as mappings from domains (inputs, stimuli) to ranges (outputs, responses). This model has been extended to address behavioral realities of network systems, where incomplete information and unpredictable behavior are the order of the day. The mathematical semantics of Flow Structures are defined in a set of theorems. For example, the Flow Structure Theorem guarantees sufficiency of the basis set of control structures for representing any flow. The Flow Abstraction/Refinement Theorem guarantees the correctness of the semantic foundations. The Flow Verification Theorem defines conditions for correctness of flows with respect to their specifications. And the System Testing Theorem defines the relationship between flows and usage models for statistical certification of system fitness for use [Linger 02a].

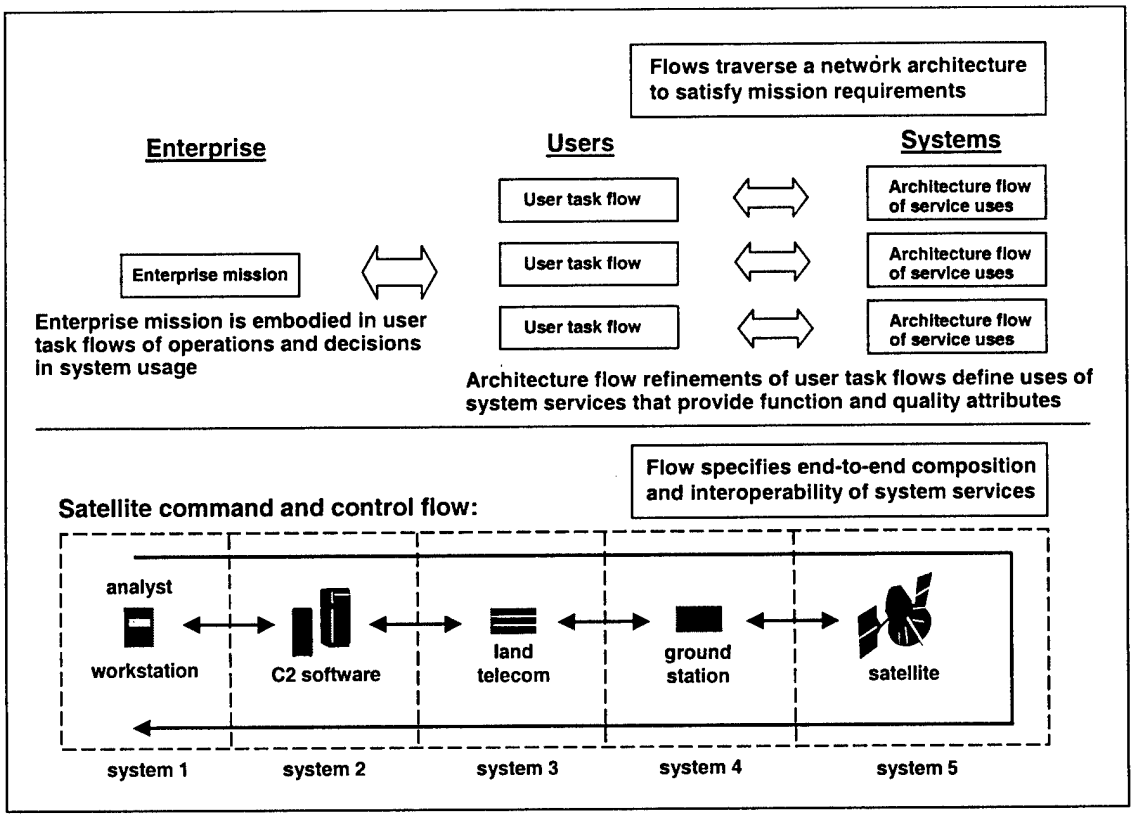


Figure 2: Flow Structure Refinement and Network Traversal

Figure 3 depicts use of Flow Structures for designing new systems and analyzing existing systems. In design, a set of flows specifies the operations a network and its services must provide. The required connectivity of services in such a FlowSet defines a sufficient logical architecture as a starting point for network design. In analysis, flows of critical tasks can be extracted from an existing architecture to reveal dependencies and potential points of failure for survivability evaluation and improvement [Moore 01].[2]

In large-scale network systems, flows can engage in extensive traversals of network nodes and communication links, where the behavior of invoked services cannot always be known and predicted. In this environment a variety of Uncertainty (Survivability) Factors must be addressed in system specification and design, including the following [Linger 02b]:

- unpredictable function: A service may be provided by COTS components of unpredictable or unknown function and reliability, and may not perform expected operations.
- compromised function: A service may have been compromised or disrupted by an intrusion or physical attack and may not be able to perform its function correctly or at all.
- modified function: A service may be modified or replaced as part of routine maintenance, error correction, or upgrade, with intentional or inadvertent modification of its function.

[2] Ibid.

- asynchronous function. A service may be used simultaneously and asynchronously by other flows, and thus produce results dependent on unpredictable history of use, both legitimate and illegitimate.

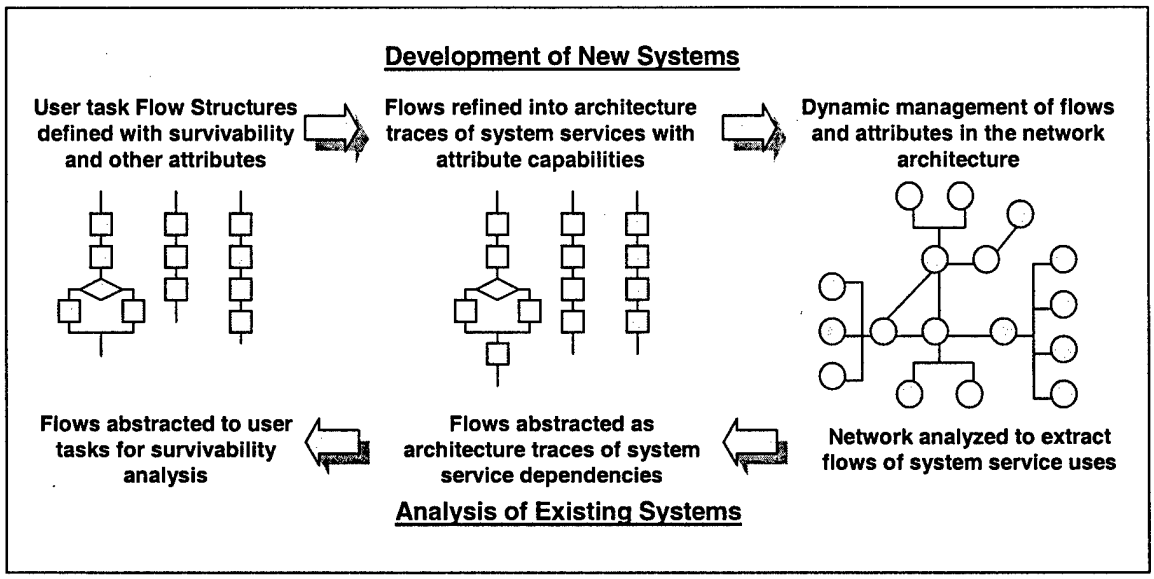


Figure 3: Flow Structure Analysis and Design

These factors are pervasive behavioral realities of large-scale, network-centric systems. Dealing with them is an enterprise risk management problem with potentially serious consequences. It is important to detect when they have occurred and take appropriate actions to continue operation. For mission-critical flows, these actions must ensure survivability no matter what adverse environments are encountered. In today's world, it is imprudent from a risk management perspective to fail to address the Uncertainty Factors in system development.

The mathematical semantics of Flow Structures are defined to support development and verification of flows for such uncertain environments as a standard engineering practice. To allow for unpredictable or unknown behavior of services, flow semantics require specification of only the processing that a flow itself performs, and not the processing of the services it invokes. That is, it is not necessary to specify details of what services do (which may be unknowable), only what a flow does with their responses. This response-based semantics ensures that flows exhibit deterministic behavior for human understanding; that is, they define unambiguous behavior, despite the underlying uncertainties and unknowns of service operations.

Flow Structure engineering requires designers to define appropriate actions by a flow for all possible responses, both desired and undesired, from critical services. This approach requires for mission survivability that the Uncertainty (Survivability) Factors be dealt with explicitly in design, thereby addressing important aspects of enterprise risk management. This is accomplished in flow design by partitioning the responses of critical service invocations into equivalence classes and providing system actions for each partition.

A high-level application of Flow Structures is depicted in Figure 4, a notional illustration of a network-centric command-and-control system whose nodes are complex systems in their own right, each with its own specified FlowSets to support. Of particular interest is use of over-arching command-and-control Flow Structures shown in the center of the figure to integrate services of individual nodes into coherent mission capabilities. These integrating flows specify requirements for the entire system and serve as stable foundations for acquisition, management, design, implementation, and operation.

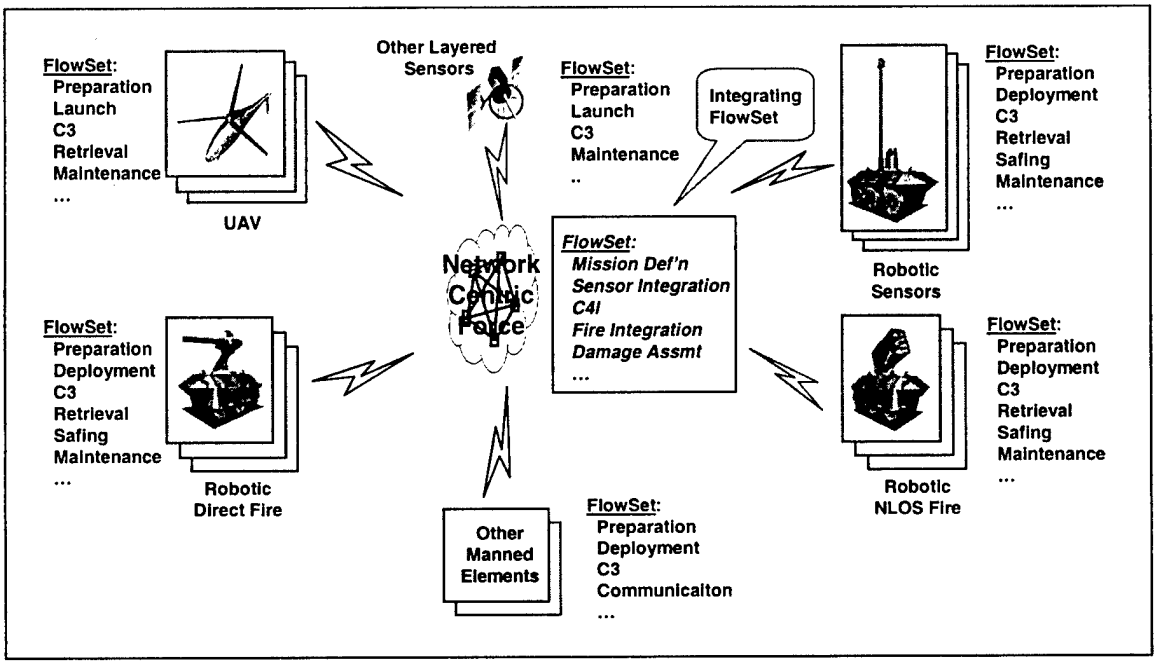


Figure 4: FlowSets for Integrating Command and Control in a Network-Centric System

4.7 Publications and Presentations

The FSQ IR&D project has produced seven publications, including an invited book chapter and three conference papers. Two additional papers dealing with the foundations of Flow Structures and Computational Quality Attributes are in preparation. FSQ technology has been presented to the Federal Aviation Administration (FAA) and a number of DoD organizations, as well as at several major conferences.

Hevner, A.; Linger, R.; Sobel, A.; & Walton, G. "Specifying Large-Scale, Adaptive Systems with Flow-Service-Quality (FSQ) Objects," 110–120. *Proceedings of the 10th OOPSLA Workshop on Behavioral Semantics*. Tampa, Florida, Oct. 14–18, 2001. Boston: Northeastern University, 2001.

Hevner, A.; Linger, R.; Sobel, A.; & Walton, G. "The Flow-Service-Quality Framework: Unified Engineering for Large-Scale, Adaptive Systems." *Proceedings of the 35th Annual Ha-*

waii International Conference on System Science (HICSS35). Hawaii, Jan. 7–10, 2001. Los Alamitos, CA: IEEE Computer Society Press, 2002.

Hevner, A.; Linger, R.; Pleszkoch, M. A.; & Walton, G. "Flow-Service-Quality (FSQ) Engineering for Specification of Complex Systems." *Practical Foundations of Business and System Specifications*, Edited by H. Kilov and K. Baclawski. New York: Kluwer, Inc., 2002.

Linger, R. *Essential Service and Sense-and-Respond Control Models* (CMU/SEI-2002-SR-007). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002 (restricted distribution).

Linger, R.; Pleszkoch, M.; Walton, G; & Hevner, A. *Flow-Service-Quality (FSQ) Engineering: Foundations for Network System Development* (CMU/SEI-2002-TN-019). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. <http://www.sei.cmu.edu/publications/documents/02.reports/02tn019.html>.

Linger, R.; Pleszkoch, M.; Walton, G; & Hevner, A. "Flow-Service-Quality Requirements Engineering For High Assurance Systems Development." *Proceedings of the International Workshop on Requirements for High Assurance Systems*, Essen, Germany, Sept. 9, 2002. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002.

Moore, A.; Ellison, R.; & Linger R. *Attack Modeling for Information Security and Survivability* (CMU/SEI-2001-TN-001). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu/publications/documents/01.reports/01tn001.html>.

5 Fusion

5.1 Purpose

The Fusion IR&D is designed to study the feasibility of multi-source data fusion for the predictive analysis of network intrusions. Multi-source data fusion has been successfully applied to the prediction of a number of other complex human-originated activities, including criminal activities and nation-state capabilities and intentions. The desire in this study is to identify and exploit data sources to gain insight into the likely targets and behaviors involved in network intrusions, particularly in *non-traditional networks*, which are networks that are temporary, ad hoc, or special purpose. By gaining this insight, a number of defensive and preventative strategies become possible, with a consequent lessening of the large amount of resources presently dedicated to network defense and recovery from intrusions.

5.2 Background

The current state of analysis of network intrusions is primarily focused on the technical details. Multiple organizations (including the CERT® Coordination Center at the SEI) produce technical advisories informing system administrators of significant new vulnerabilities and of the exploitation of these vulnerabilities in network and host intrusions. One limitation of the purely technical approach is that it contains little information with respect to the motivations and triggering events for intrusions, and, as a result, may provide little basis for prioritization of the vulnerability or linking to current operational plans of the receiving organizations. Many of the analyses that focus on motivations and triggering events (including much of those produced by the National Intelligence Community) focus on the socio-political aspects of network threats, but provide little in the way of technical insight, and may overstate the ease of compromise.

Disturbingly, there have been several attempts at fusion analyses in the past year or so that have provided only alarmist or surface detail, and have jumped to disturbing and poorly justified conclusions. The Institute for Security Technology Studies at Dartmouth University recently published a report titled "Cyber Attacks During the War on Terrorism: A Predictive Analysis."[3] This report documented a number of nuisance-level network intrusions (focusing

® CERT and CERT Coordination Center are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

3 Available at http://www.ists.dartmouth.edu/ISTS/counterterrorism/cyber_attacks.htm

mainly on Web site defacements), then leaped to the poorly justified conclusion that cyber attacks on U.S. infrastructures were imminent. The Gartner Group conducted a multifaceted study of cyberterrorism titled "Digital Pearl Harbor,"[4] but this study was flawed by apparently limited investigation into the defenses employed in the infrastructures studied.

Based on this survey of previous efforts, admittedly limited to the open literature, full understanding of networked threats require a comprehensive and mission-oriented predictive analysis method using fusion of data from multiple disciplines. This IR&D project developed such a method and applied it to non-traditional networks. The nontraditional network focus was selected to both reinforce the mission orientation and to limit the scope of the analysis to a tractable level.

5.3 Approach

One starting point for the development was the recognition that the goal is not necessarily to protect a particular host or network, but to support the mission associated with that network. Partially, this recognition stems from the definition of survivability. Further, in working with non-traditional networks, the concern is far more on the usage of the network (the mission) than on the structure and implementation of the network. This recognition forces orientation of the analysis to the mission, rather than to the technical aspects of protocols, services, and implementation. The mission orientation in turn motivates analysis to view the entire threat environment, rather than focusing primarily on organizational or technical vulnerabilities. Missions are activities of an organization, and are hosted by or facilitated through network services, but are not themselves network services. Considering the threat environment requires positing motivation and identifying potential trigger events. These trigger events may be technical (e.g., the release of a new exploit method), but are also possibly social (the tangling of two skaters in an Olympic race), political (an international incident), or economic (the release of a new product or service). These motivations or triggering events are as important in considering the threat as the available tools and methods of compromise. Once these are identified, then the architecture and operation of the network with respect to the mission may be analyzed to understand and define the potential impacts of various courses of attack. In performing this analysis, consideration must be given to the impact of the analytic methodology itself, in varying the degree of attention paid to potential attacks. Particularly in a temporary network, there are not sufficient time and resources to respond to all potentially threatening network traffic, so prioritization must be given to mission-threatening traffic. Since the analysis will more fully define "mission threatening traffic," it will change this prioritization.

While this methodology builds in concept from a variety of sources, the innovative aspect is the focus on nontraditional networks. The identified consideration of cascade effects between

[4] Discussed at http://www3.gartner.com/2_events/audioconferences/dph/dph.html

the physical and cyber realms and the work with temporary or ad hoc networks is specifically a new aspect of this work. As the study progressed, it became clear that this analysis has implications both in the protection of missions and in the investigation of crimes against organizations. This dual application is also an innovative aspect of this work.

While this study has laid out the basic structure of the analysis and employed it with specific pilot applications, there remain several issues to be resolved. One is identification of the degree to which a non-traditional network may be affected by insider and non-insider attacks. Information on insider attacks is fairly difficult to obtain, and the distinguishing characteristics of insider vs. non-insider attacks remain obscure. Another unresolved issue is the method of validating the degree of connectivity at the mission level between non-traditional networks. Work has been done at a number of organizations on validating connectivity at the protocol levels, but the connections between missions have been relatively unexplored. Another unresolved issue is identifying the type of efficacy of existing protective measures in the infrastructure. Matching applicability of a protective measure in dealing with a specific attack is difficult enough; considering the operational efficacy in terms of how the impacts of the attack are mitigated by a control is more difficult still.

There are several direct impacts from the success of this analysis methodology:

- greater protection of critical infrastructure, by supporting understanding of impacts and prioritization of response
- greater protection of network-enabled events, in dealing with the network in a mission-oriented manner
- closer focus by investigative agencies on realistic risks
- improved training of protective and investigative personnel

5.4 Collaborations

The U.S. Secret Service and Phil Williams, a visiting scientist from the University of Pittsburgh, collaborated with Tom Longstaff, Casey Dunlevy, and Tim Shimeall of the SEI's CERT Analysis Center on this project.

5.5 Evaluation Criteria

The following criteria were used to evaluate this project:

- publication in reviewed conferences and journals
- identification of and interest by potential sponsors
- facilitation of application to a variety of areas of work

5.6 Results

The results of the project are as follows:

- We have developed the capability to provide realistic threat evaluations for temporary or ad hoc networks, based on fusion of technical, social, political, and (to a lesser degree) economic data.
- The methodology for providing this threat evaluation has been piloted at multiple events.
- These threat predictions have been used to develop guidelines for the protection of these networks, increasing the security of the events associated with these networks.
- These threat predictions have been used to provide content for training courses for protection agencies.
- We have developed the capability to provide realistic threat predictions obtained in cyber contexts to physical world environments, both by studying the threats arising from networked control systems and from infrastructures that rely heavily on networked information systems.
- The methodology for providing this threat evaluation has been published in both journal and conference form (multiple conferences).
- This threat analysis has been applied to both protective and investigative guidelines associated with temporary and permanent networked infrastructures.
- This threat analysis has been piloted in coursework at Carnegie Mellon University and beyond.

5.7 Publications and Presentations

Shimeall, T. J.; Dunlevy, C. J.; & Williams, P. "Models of Information Security Trend Analysis." SPIE Aerosense Law Enforcement Technologies Conference, Orlando, FL, April 2002.

Dunlevy, C. J.; Shimeall, T. J.; & Williams, P. "Cyber Intelligence Analysis." *Contemporary Security Policy 23*, 2 (August 2002).

Shimeall, T. J. & Williams, P. "Cyber Security Threats to the Financial Industry." Presentation to Federal Financial Institution Examination Council Information Systems and Technology Conference, Pittsburgh, PA, August 2002.

6 Open Source Software (OSS)

6.1 Purpose

The purpose of the Open Source Software (OSS) IR&D effort is to determine whether OSS is indeed the "silver bullet" that some have called it. OSS may be the next innovation or great leap forward in the way the software community develops software; however, it may also be a dud—and without a crystal ball it is hard to tell at this time. So until it becomes possible to see into the future, we (i.e., the software community) can only analyze the current situation to gain an understanding of what OSS is, how it is developed, and how it is contributing to the way we develop software, and to learn where we can apply OSS in the overall science of software engineering.

6.2 Background

6.2.1 What Is Open Source Software?

The term *open source software* at the most basic level simply means software for which the source code is open and available as defined below:

- *open*—The source code for the software can be read (seen) and written (modified). Further, this term is meant to promote the creation and distribution of derivative works of the software.
- *available*—The source code can be acquired either free of charge or for a nominal fee (e.g., media and shipping charges or online connection charges).

Today, making source code *available* can be as simple as posting the code on the World Wide Web (WWW) or posting it in an online newsgroup. Making the software *open* is also simple—place no restrictions on how the software is actually used or by whom.

Others have gone to great lengths to define OSS. In fact an entire group, the Open Source Initiative (OSI), formed and established the Open Source Definition (OSD) [OSI 01a].

6.2.2 History of OSS

Software source code that is open and available has been around since the earliest days of modern computing [Arief et al. 01]. Feller and Fitzgerald provide a good historical account of open and available source code from the 1940s to the contemporary advocacy campaign of the OSI [Feller et al. 02]. In their book, Feller and Fitzgerald acknowledge some of the earliest code-sharing activities between scientists working on some of the earliest computers, such as the Electronic Numerical Integrator and Calculator (ENIAC); formalized groups that share software, such as the Project for Advancement of Coding Techniques (PACT); and published articles including source code, such as "Algorithms" in the *Communications of the ACM* [Feller et al. 02, Leonard 00].

Many trace the beginnings of the modern open source movement back to more recent projects including (but not limited to) the University of California at Berkeley's software distribution of Unix (BSD Unix) and later to the formation of the GNU ("GNU's Not Unix") Project and the establishment of the Free Software Foundation (FSF) by Richard Stallman. The purpose of the GNU Project was to create a free version of Unix and Unix tools not impaired by the restrictions of licensing and distribution.

6.2.3 OSS as a Silver Bullet

How OSS came to be labeled as a silver bullet is a matter of opinion and speculation. Much of any opinion (including ours) would be based on the attention that OSS has received in the mainstream press. Growth in the software industry, fueled by the explosion of e-commerce and dot-com companies through the 1990s —all in response to the overwhelming acceptance of and interest in the World Wide Web—warranted such coverage. The list of companies that grew from literally nothing to companies whose stock was worth millions of dollars on Wall Street gave credence to what then appeared as a phenomenon whose potential was limited only by those investors who were willing to dump money into dot-com companies.

Such tremendous starts, which garnered national and worldwide attention, brought OSS and the companies that intended on making a business from OSS (such as RedHat and VA Linux) into the mainstream.

6.2.4 Current State of OSS

There is the assumption that OSS, being under constant peer review by developers around the world and around the clock, must therefore be of high quality, specifically

- (more) reliable
- (more) robust
- (more) secure (or no security through obscurity)

Raymond makes two basic arguments that support this notion [Raymond 99]. The first is that hackers (OSS developers), knowing in advance that others will see their code, will be more likely to write the best code that they can possibly write—out of fear of being embarrassed before the community for writing anything less.

The second argument is asserted as *Linus's Law*: "Given enough eyeballs, all bugs are shallow" [Raymond 99]. Again the notion is that because there are thousands of developers reviewing OSS code, 24x7x365, a flaw in the code will be obvious (hence shallow) to someone (who will either report or fix it). Based on those two premises, the common community belief is that OSS is of higher quality, in the sense that it applies to all of the '*ilities.*[5]

In fact there is open and available software, which is good software and, by many measures, high-quality software. The question posed in this OSS IR&D study is whether all OSS should share the same status as high-quality software.

Beyond the qualities (perhaps perceived qualities) of OSS, other notions predominate the commonly accepted perceptions of OSS, which are addressed in this IR&D study and listed below:

- Having the source code yields greater control of that open source code as well as the system in which that code is included.
- OSS is notoriously lacking in quality documentation (if it has any at all).
- There is a world's worth of programmers sitting around eagerly awaiting the next piece of source code to write.
- Programmers in the OSS community are a group of mavericks working in an unorganized, haphazard, ad hoc fashion.

6.3 Approach

We attempted to understand the OSS phenomenon by actually getting involved in or researching several efforts/events. There were five such studies:

- AllCommerce—an e-commerce storefront solution
- Apache—an open source Web server
- Enhydra™—a Java-based application server
- NAIS—a NASA-operated Web site that switched from Oracle to MySQL
- Teardrop—a successful Internet attack affecting OSS and CSS

---

5 The term 'ilities is often used to refer to various properties of systems and components in general, such as scalability, reliability, security, and adaptability.

™ Enhydra is a trademark of Lutris Technologies, Inc.

The purpose in selecting these specific OSS projects was to take varying perspectives of OSS, in terms of software development, the products themselves, and users.

The AllCommerce case study focused on software development in the OSS paradigm. A member of the SEI technical staff got involved in the process of hacking the product to discover bugs and add new features to the product. The express purpose of this case study was to obtain firsthand experience in working on an OSS product from the *inside*, that is, to learn the process by which changes are actually proposed, tracked, selected/voted on, and accepted.

The Apache case study takes an academic, research perspective (actually the result of a doctoral thesis) of the OSS-development process. This case study looked at the individual contributions made to the Apache Web server over the past five years and whether that contributor was from core or non-core Apache developers.

From a purely product-centric perspective, the Enhydra case study focused on the qualitative aspects of an OSS product and looked at coding problems found in the product by conducting a critical code review.

The NAIS case study, which focused on the end user, looked at a real application developer who switched from a commercially acquired software product to an OSS product. Specifically, this case study examined how and why that particular OSS product was selected, the degree to which the application developer was engaged with the OSS development community, and the level of satisfaction that the NAIS had with the selected OSS product.

Finally, the Teardrop case study looked into one of the predominant axioms of OSS: that OSS is more secure than software developed under more traditional means. This case study takes apart one of the most successful distributed denial-of-service (DDoS) attacks, and looks at the role that OSS played in the propagation of that attack on CSS and the response by the OSS community.

6.4 Collaborations

Scott Hissam, Charles B. Weinstock, and Daniel Plakosh, members of the SEI technical staff, have collaborated with the following organizations on this project:

- Collab.NET, leaders in collaborative software development technologies and methodologies
- AllCommerce; other unnamed OSS developers working on AllCommerce
- Apache; Jayatirtha Asundi from Carnegie Mellon's Software Industry Center
- NAIS & NASA; they kindly provided responses to our questionnaire.

6.5 Evaluation Criteria

The evaluation criteria for this work, all of which were met successfully, were as follows:

- development of a technical report. This technical report is the culmination of the research conducted and its findings. It was published in November 2001 after being externally reviewed by Tim O'Reilly (O'Reilly & Associates), Peter G. Neumann (SRI International Computer Science Laboratory), and Joseph Feller (University College Cork, Ireland).
- publication of an OSS paper in at least one major journal. The paper titled "Trust and Vulnerability in Open Source Software" was published in the February 2002 issue of *IEEE Proceedings—Software*.
- leadership in the community. As part of this study, members of the team held a Birds-of-a-Feather (BoF) session at the O'Reilly Open Source Convention, participated in the 1st International Workshop on Open Source Software Development at the 23rd International Conference on Software Engineering, and organized and conducted the Second Workshop on Open Source Software Development at the 24th International Conference on Software Engineering. Also as a measure of leadership success, members of this team provided press interviews to *Federal Computer Week*, *Government Computer News*, *New York Times Online*, and *Business Week Online*. Finally, the team gave a presentation to the President's Information Technology Advisory Committee (PITAC) subpanel on OSS.

6.6 Results

Instances of successful OSS products such as Linux, Apache, Perl, sendmail, and much of the software that makes up the backbone of today's Web are clear indications that successful OSS activities can occur often. Certain conditions appear to be key for such success; specifically, OSS must

- **be a working product**. Looking back at many of the products, especially Apache and Linux, none started in the community as a blank slate. Apache's genesis began with the end of the National Center for Supercomputing Applications (NCSA) Web server. Linus Torvalds released Linux version 0.01 to the community in September 1991. Just a product concept and design in the open source community has a far less likely chance of success. A prototype, early conceptual product, or even a *toy* is needed to bootstrap the community's imagination and fervor.
- **have committed leaders**. Equally as important is a visionary or champion of the product to chart the direction of the development in a (relatively) forward direction. Although innovation and product evolution are apt to come from any one of the hackers in the development community, at least one person is needed to be the *arbiter of good taste* with respect to the product's progress. This is seen easily in the Apache project (regarding the Apache Foundation).
- **provide a general community service**. This is perhaps the closest condition to the business model for commercial software. It is unlikely that a commercial firm will bring a product to market if there is no one in the marketplace who will want to purchase that product. In the open source community, the same is also true. Raymond points out a few valuable lessons:

- *"Every good work of software starts by scratching a developer's personal itch."*
- *"Release early, release often. And listen to your customers."*
- *"To solve an interesting problem, start by finding a problem that is interesting to you"* [Raymond 99].

- **be technically *cool*.** You are more likely to find an OSS device driver for a graphics card than an accounting package. Feller and Fitzgerald categorized many of the open source projects in operation, noting that a high percentage of those were Internet applications (browsers, clients, servers, etc.), system and system-development applications (device drivers, code generators, compilers, and operating systems/kernels), and game and entertainment applications [Feller et al. 02].
- **have developers who are also its users.** The characteristic that is perhaps most indicative of a successful OSS project is that the developers are also the users. Typically this is a large difference between OSS and commercial software. In commercial software, users tend to convey their needs (i.e., requirements) to engineers who address those needs in the code and then send the software back to the users. A cycle ensues with users conveying problems and the engineers fixing and returning the code. However in OSS, it is more typical that a skilled engineer would rather repair the problem in the software and report the problem *along with the repair* back to the community. The fact that OSS products are technically *cool* explains why many of the most popular ones are used typically by the developer community on a day-to-day basis. (Not many software developers we know use accounting packages!)

The SEI views OSS as a viable source of components from which to build systems. However, we are not saying that OSS should be chosen over other sources simply because the software is open source. Rather, like commercial off-the-shelf (COTS) software and closed source software (CSS), OSS should be selected and evaluated on its merits. To that end, the SEI supports the recommendations of the President's Information Technology Advisory Committee (PITAC) subpanel on OSS to remove barriers and educate program managers and acquisition executives and allow OSS to compete on a level playing field with proprietary solutions (such as COTS or CSS) in government systems [PITAC 00].

Adopters of OSS should not enter into the open source realm blindfolded but should know the real benefits and pitfalls that come with OSS. The fact that OSS is open means that everyone can know the business logic encoded in the software that runs those systems. That, in turn, means that anyone is free to point out and potentially exploit the vulnerabilities with that logic—and anyone could be the altruistic OSS developer or the cyber terrorist. Furthermore, having the source code is not necessarily the solution to all problems: without the wherewithal to analyze or perhaps even to modify the software, having it makes no difference.

6.7 Publications and Presentations

Hissam, S.; Weinstock, C.; Plakosh, D.; & Asundi, J. *Perspectives on Open Source Software* (CMU/SEI-2001-TR-019, ADA401728). Pittsburgh, PA: Software Engineering Institute,

Carnegie Mellon University, 2001.
<http://www.sei.cmu.edu/publications/documents/01.reports/01tr019/01tr019title.html>.

Weinstock, C. "PITAC Cyberterrorist" Presentation. The 2000 SEI Symposium. Washington, D.C., September 18-21, 2000.

Hissam, S. & Weinstock, C. "Open Source Software: The Other Commercial Software." Making Sense of the Bazaar: 1st Workshop on Open Source Software Engineering. *Proceedings of the 2001 International Conference on Software Engineering*. Toronto, Ontario, Canada, May 12-19, 2001.

Hissam, S. & Weinstock, C. "Trust & Vulnerability in Open Source Software" Presentation to the President's Information Technology Advisory Committee (PITAC) Panel on High Performance Computing, subpanel Open Source Software. San Diego, California, March 24, 2000.

Hissam, S. & Weinstock, C. "Perspectives on Open Source Software" Presentation. Birds-of-a-Feather Session at the O'Reilly Open Source Software Convention. Monterey, California, July 18, 2001.

Hissam, S.; Plakosh, D.; & Weinstock, C. "Trust and Vulnerability in Open Source Software." *IEE Proceedings—Software 149,* 1 (February, 2002): 47-51.

7 Predictable Assembly from Certifiable Components (PACC)

7.1 Purpose of the PACC IR&D

Significant economic and technical benefits accrue from the use of pre-existing and commercially available software components to develop new systems. However, variable component quality, combined with hidden "black box" component behavior, has forced system developers to rely upon extensive prototyping just to establish the feasibility of using a component in a particular assembly; predictability is a distant possibility. Many of the benefits of software component technology evaporate in the presence of high design uncertainty and low consumer trust in components.

The predictable assembly from certifiable components (PACC) project was initiated to determine whether, and how, the twin challenges of design predictability and component trust could be addressed. A software development activity is predictable if the runtime behavior of an assembly of components can be predicted from the known properties of components and if these predictions can be objectively validated. A component is certifiable if these known properties can be ascertained and validated by independent third parties.

As shown in Figure 5, PACC addresses the twin challenges of predictable assembly and trusted components by relating the two problems. In PACC, we say that the only component properties that are worth trusting are those that support reasoning about design-level (assembly-level) properties. The processes of identifying the properties of components whose values must be certified, and defining practical means of certifying these properties, can be mutually informing.

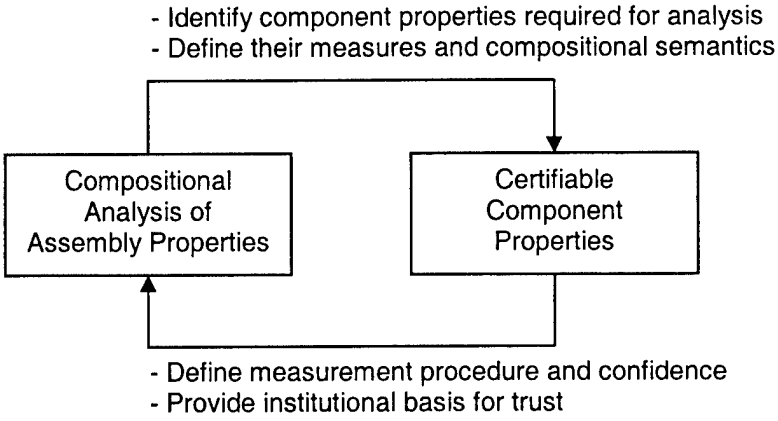


Figure 5: Twin Challenges Addressed by PACC

7.2 Background: Software Component Technology

Software components have been available in the commercial marketplace since the late 1970s, but it has only been in the 1990s that the market potential of and the corresponding demand for software components have been manifested. The result has been a dramatic transition from nearly exclusive emphasis on top-down, custom development to bottom-up, component-based development. The SEI's COTS-Based Systems Initiative was established to address the challenges that have attended this industry-wide transition.

Many of the technical challenges of component-based development arise from the same market pressures that gave rise to software components in the first place. Component vendors are forced to compete for a limited consumer base, and early profits from innovation shrink as competitors introduce components with similar capabilities. This competition results in a component market that is dominated by innovative and fast-changing features. While this has produced many benefits for consumers, it has also vastly complicated the task of the system integrator—components often do not fit together, and maintaining fit over new component versions is problematic. The SEI has developed methods to meet these new challenges [Wallnau 01], but there is no denying that these methods are reactive; the fundamental technical challenges of component integration and maintenance remain.

Ironically, the component market is also responding to these market-induced challenges. A new spate of commercial offerings, under the broad rubric of software component technology, has emerged—in no small part spurred by the Java and Internet revolutions. Component technology promises, and to a large extent succeeds, in reducing the complexity of component integration. However, as it exists today, component technology is more marketing than innovation. The major technical elements of component technology—for example, interface types, encapsulation, and specialized runtime environments—have existed for many years. What is significant about component technology is that information technology (IT) produc-

ers and consumers have been rapidly adopting it in the form of Sun Microsystems' EJB™ (Enterprise Java Bean) and Microsoft's COM™ and .NET, to name a few [Bass 01].

However, while current-generation component technologies address various syntactic and mechanistic aspects of component assembly, more complex forms of behavioral assembly have not been addressed. It is precisely these aspects of assembly that pose the most pressing challenges to the Department of Defense (DoD) and other producers of complex, mission-critical systems. As software systems have become more complex, these limitations of existing component technologies have become apparent.

Significant research has been undertaken in the area of software architecture to support compositional reasoning about critical system-level behavioral properties, such as liveness, safety, and performance (e.g., [Magee 97]). However, the results of this research have not been translated for use where components are implementations rather than just design abstractions; the properties of an assembly of component implementations remain mysterious until testing time. Efforts to develop a basis for trusted components have yielded indifferent results. Current best practice is to specify interface contracts (e.g., [Meyer 00]), but this is inadequate for the critical system-level properties mentioned earlier. Measures of component quality are either directed at development processes (e.g., [UL 98]), or are objective but are non-composable (e.g., [Voas 00]); i.e., measures of component properties can not be combined (*composed*) to produce assembly properties (see [Stafford 02]).

7.3 Approach: Predection-Enabled Component Technology

Our approach to predictable assembly is to augment component technologies with sound analysis and prediction technologies. We refer to the resultant augmentation as a *prediction-enabled component technology* (PECT). This marriage of component and analysis technology makes sense:

- Every analysis model is valid only with respect to assumptions about the execution environment of an end application. For example, a performance model will likely depend upon assumptions pertaining to scheduling policy, process or thread priority, concurrency, resource management policies, and many other factors. These assumptions can be treated as design and implementation constraints, and made explicit, supported, and enforced by component technology. That is, assemblies of components can be rendered *analyzable by design and construction.*
- Analysis models refer to (are parameterized by) the properties of components being modeled. We refer to these properties as *analytic properties* and refer to the set of these as the component's *analytic interface.* An explicit, well-defined analytic interface provides an opportunity for certifying just those component properties that support engineering analysis. This, in turn, provides a value proposition for certified, or *trusted*, component properties.

- Component technology is *par excellence* a means of packaging and deploying software technology. In addition, it is being adopted by industry. On a very practical level, we view component technology as a readily available distribution channel for packaging and deploying predictable assembly from certifiable components.

The last and defining element of our approach is that the packaging of a PECT is not complete until its predictive powers have been validated. Our objective is that each PECT be described by an objective, bounded confidence interval that is backed by mathematical and empirical evidence. We exclude from our purview any analysis technology that can not, in principle or practice, support such validation.

The user-level workflow for PECT is shown in Figure 6. Note that the *Validate prediction* step is distinct from the validation of the PECT itself. In this workflow, validation refers to a spot check of the assembly against a prediction. We have taken liberties with Unified Modeling Language (UML) by permitting multiple 'no' paths on conditional branches. The terms *virtual assembly*, *concrete assembly*, and *analytic assembly* are found in the conceptual schema, which identifies the key terms and their relationships. This conceptual schema is shown in Figure 7.

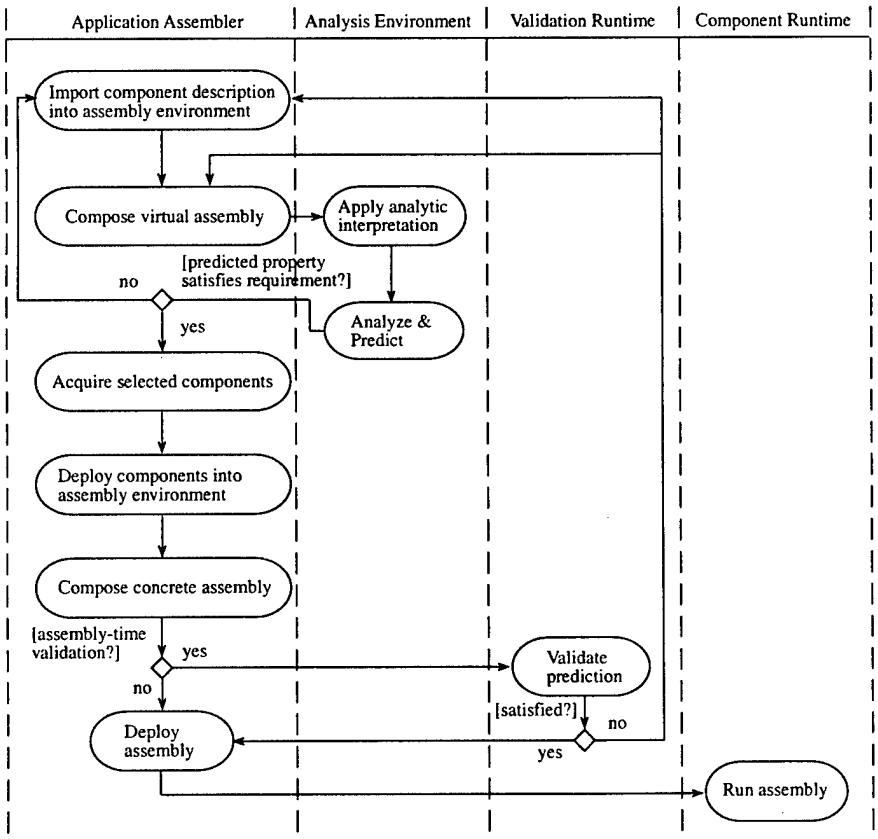


Figure 6: The User-Level Workflow for PECT

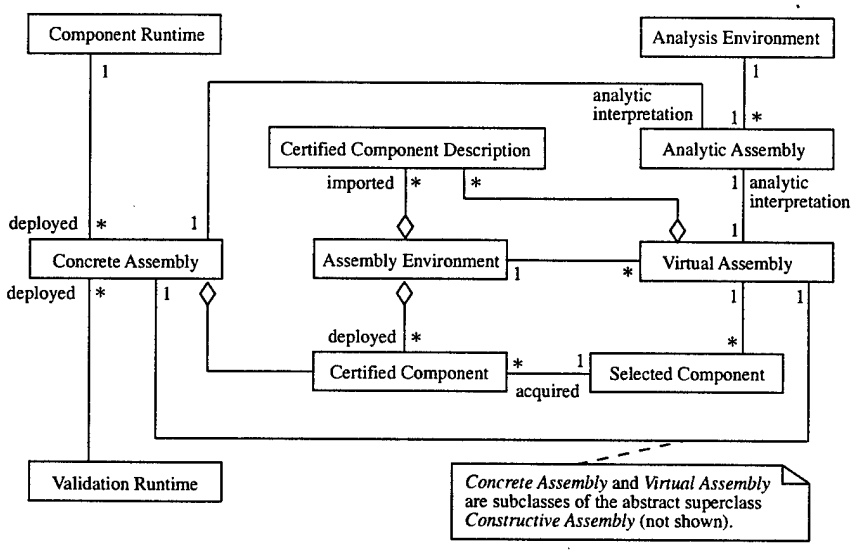


Figure 7: The PECT Conceptual Schema

7.4 Collaborations on Predictable Assembly

Scott Hissam, John Hudak, James Ivers, Mark Klein, Gabriel Moreno, Linda Northrop, Daniel Plakosh, Judy Stafford, Kurt Wallnau, and Bill Wood are the members of the SEI technical staff who have contributed to this project.

A major objective of our research was to establish a forum for researchers working in areas closely related to PACC to share results. In collaboration with the International Conference on Software Engineering (ICSE) series, the SEI has hosted two workshops on component-based software engineering that focused directly on the topics of predictable assembly and component certification [Crynkovic 01, Crynkovic 02a], and one invitation-only workshop to develop model problems for PACC research [Crynkovic 02b]. These results have netted, intangibly, increased collaboration in the research community; more tangibly, the SEI has co-edited one special issue of the *Journal of Systems and Software* devoted to predictable assembly[6] and is in the process of producing a follow-on special issue based on the results of the 2002 ICSE Workshop.

The SEI has also benefited from industrial funding. In 2001-2002, Asea Brown Boveri, Ltd. (ABB) provided $250,000 to demonstrate the use of PECT for critical infrastructure systems—substation automation systems within the domain of power transmission and control. For this work, ABB and Mälardalen University, Sweden, co-sponsored a PhD candidate, Magnus Larsson, as a resident affiliate of the SEI. ABB has doubled the funding to $500,000

[6] "Component-Based Software Engineering: Component Certification and System Prediction." *Special Issue of the Journal of Systems and Software*, Elsevier Science (to appear).

for 2002-2003 to encompass industrial robotics and to introduce more stringent feasibility requirements (see Section 7.6, Results).

7.5 Evaluation Criteria

The evaluation criteria for PACC reflected its practical but rigorous agenda by posing the following questions:

- Can component properties be reliably obtained, and used compositionally, to predict the properties of assemblies of components?
- Can we assign objective measures of confidence to component properties and make design (assembly-level) predictions based on them?
- Can the technology that enables PACC (that is, PECT) be packaged to hide the complexity of the embedded theories of system behavior?
- Can a systematic method be defined for designing, validating, and deploying PECT for complex, real-world application domains?
- Can all of the above be done in a way that is economically and practically achievable?

The PACC IR&D effort officially ends in September 2002, when it becomes an *Emerging Initiative*. This transition reflects the affirmative answer to all of the above criteria except, perhaps, criterion 5. Establishing this last criterion is the work of the newly formed PACC Initiative. We are confident that the new initiative can expand on the technical foundations reflected in the other criteria, and, ultimately, apply and amplify PACC ideas in broad industry practice.

7.6 Results

The major results of the PACC IR&D are discussed in terms of the above evaluation criteria. The following terse summaries are developed in depth in published papers and technical reports. A high-level overview of the technical concepts of PACC and PECT can be found in an SEI technical report being developed by Kurt Wallnau,[7] while details of the industrial feasibility demonstrations are described in an SEI technical report being developed by Scott Hissam et al.[8]

---

[7] Wallnau, K. *Volume III: Technical Concepts of Predictable Assembly from Certifiable Components* (CMU/SEI-2002-TR-030). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University (in development).

[8] Hissam, S.; Hudak, J.; Klein, M.; Larsson, M.; Moreno, G.; Northrop, L.; Plakosh, D.; Stafford, J.; Wallnau, K.; & Wood, W. *Feasibility Demonstrations in Predictable Assembly* (CMU/SEI-2002-TR-031). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University (in development).

Can component properties be reliably obtained, and used compositionally, to predict the properties of assemblies of components? The project demonstrated that timing properties of software components can be reliably obtained in several different environments: the component development environment, third-party certification environment, or application assembly environment [Hissam 02, Moreno 02, Stafford 01]. Techniques for obtaining measures of empirical component properties, such as time, complement research on the subject of transmitting component formal properties that have been established by proof-theoretic means [Necula 96].[9]

Can we assign objective measures of confidence to component properties and make design (assembly-level) predictions based on them? The project developed a rigorous process for demonstrating the empirical validity of component properties and predictions, and proposed conventions for standard statistical labeling of properties and predictive models [Moreno 02]. We explored, with Carnegie Mellon University (CMU) Sloan Software Center and the CMU Institute for Software Research, how these standard labels might enable insurance underwriting of software components [Li 02].

Can the technology that enables PACC (that is, PECT) be packaged to hide the complexity of the embedded theories of system behavior? A substantial emphasis of the PACC work has been to reduce the complexity of PECT experienced by consumers through *zero-programming assembly*, where the major abstractions for application development are components and connectors. We developed the *Pin Component Model* to demonstrate and help explore the potential of zero-programming assembly. A report on a specification language and composition semantics for Pin can be found in an SEI report being developed by J. Ivers and others.[10]

Can a systematic method be defined for designing, validating, and deploying PECT for complex, real-world application domains? A substantial collection of processes have been defined to design and validate a PECT, and they have been applied in our case studies. These processes specify *deliverables*, *artifacts*, *workers*, their *activities,* and *workflow*. The project also developed technology infrastructure to support some of these workflows (for example, those concerning measurement and validation of component measures and assembly prediction).

Can all of the above be done in a way that is economically and practically achievable? Although we can not claim to have demonstrated that PECT is practical and economical when scaled, preliminary results are encouraging. Besides speculations about component insurance

---

9 <http://www.cs.princeton.edu/sip/projects/pcc/> is a good source of links on this topic.

10 Ivers, J.; Sinha, N.; & Wallnau, K. *Syntax and Semantics for a Simple Composition Language: Interim Report* (CMU/SEI-2002-TN-026). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University (in development).

underwriting [Li 02, UL 98], there is also apparent interest in applying a product line approach to PECT [Larsson 02]. We believe the economic need for trusted components and predictable assembly will only increase over time, and PECT will be correspondingly attractive as an approach to PACC.

7.7 Publications and Invited Presentations

Bachmann, F.; Bass, L.; Buhman, C.; Comella-Dorda, S.; Long, F.; Robert, J., Seacord, R.; & Wallnau, K.; *Volume II: Technical Concepts of Component-Based Software Engineering* (CMU/SEI-2000-TR-008, ADA 379930). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. <http://www.sei.cmu.edu/publications/documents/00.reports/00tr008.html>.

Bass, Len; Buhman, Charles; Comella-Dorda, Santiago; Long, Fred; Robert, John; Seacord, Robert; & Wallnau, Kurt. *Volume I: Market Assessment of Component-Based Software Engineering Assessments* (CMU/SEI-2001-TN-007, ADA 395250). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu/publications/documents/01.reports/01tn007.html>.

Crynkovic, I., Schmidt, H.; Stafford, J.; & Wallnau, K. (Eds). *Anatomy of a Research Project in Predictable Assembly* (White Paper). <http://www.sei.cmu.edu/pacc/CBSE5/CBSE5_whitepaper.pdf>.

Crynkovic, I.; Schmidt, H.; Stafford, J.; & Wallnau, K. (Eds). *Proceedings of the 4th ICSE Workshop on Component-Based Software Engineering: Component Certification and System Prediction.* <http://www.sei.cmu.edu/pacc/CBSE4-Proceedings.html>.

Crynkovic, I.; Schmidt, H.; Stafford, J.; & Wallnau, K. (Eds). *Proceedings of the 5th ICSE Workshop on Component-Based Software Engineering: Component Certification and System Prediction.* <http://www.sei.cmu.edu/pacc/CBSE5/CBSE5-Proceedings.html>.

Crynkovic, I.; Schmidt, H.; Stafford, J.; & Wallnau, K. *Workshop Summary of the 5th ICSE Workshop on Component-Based Software Engineering: Benchmarks for Predictable Assembly, in Software Engineering Notes* (to appear).

Hissam, S.; Hudak, J.; Klein, M.; Larsson, M.; Moreno, G.; Northrop, L.; Plakosh, D.; Stafford, J.; Wallnau, K.; & Wood, W. *Feasibility Demonstrations in Predictable Assembly* (CMU/SEI-2002-TR-031). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University (in development).

Hissam, S.; Moreno, G.; Stafford, J.; & Wallnau, K. "Enabling Predictable Assembly, Component-Based Software Engineering: Component Certification and System Prediction." *Special Issue of the Journal of Systems and Software.* Elsevier Science (to appear).

Hissam, S.; Moreno, G; Stafford, J.; & Wallnau, K. "Packaging Predictable Assembly." *Proceedings of the 1st IFIP/ACM International Working Conference on Component Deployment.* Berlin, Germany, June 2002 <http://www.sei.cmu.edu/staff/kcw/cd2002.pdf>.

Hissam, S.; Moreno, G; Stafford, J.; & Wallnau, K. *Packaging Predictable Assembly with Prediction-Enabled Component Technology* (CMU/SEI-2001-TR-024, ADA 3399793). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu/publications/documents/01.reports/01tr024.html>.

Ivers, J.; Sinha, N.; & Wallnau, K. *Syntax and Semantics for a Simple Composition Language: Interim Report* (CMU/SEI-2002-TN-026). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University (in development).

Li, P.L.; Shaw, M.; Stolarick, K.; & Wallnau, K. "The potential for synergy between certification and insurance." *Proceedings of the First International Workshop on Software Reuse Economics* (held in conjunction with the Seventh International Conference on Software Reuse). Austin, Texas, April 16, 2002. <http://www.sei.cmu.edu/staff/kcw/icsr02.pdf>.

Moreno, G; Hissam, S.; & Wallnau, K. "Statistical Models for Empirical Component Properties and Assembly-Level Property Predictions: Toward Standard Labeling." *Proceedings of the 5th ICSE Workshop on Component-Based Software Engineering.* Orlando, Florida, May 2002. <http://www.sei.cmu.edu/pacc/ CBSE5/Moreno-cbse5-final.pdf>.

Sinha, N.; Ivers, J.; Clarke, E.; & Wallnau, K. "Compositional Validation of Component Technology" (Invited Talk). Navy Critical Infrastructure Production University Research Initiative Meeting. July 11-12, 2002, Harper's Ferry WV.

Stafford, J. "Bridging the Gap Between Software Architecture and Component Technology" (Invited Talk). Politecnico di Milano, Dipartimento di Elettronica ed Informazione Colloquium. June 24, 2001.

Stafford, J. "Bridging the Gap Between Software Architecture and Component Technology" (Invited Talk). Mälardalen University. August 31, 2001.

Stafford, J. "Predictable Assembly from Certifiable Components" (Invited Talk). Component-Based Design of Safety Critical Vehicular Systems (SAVE) Workshop. Mälardalen University April 12, 2002.

Stafford, J. & McGregor, J. "Issues in Predicting the Reliability of Composed Components." *Proceedings of the Fifth ICSE Workshop on Component-Based Software Engineering.* Orlando, Florida, May 2002. <http://www.sei.cmu.edu/pacc/CBSE5/StaffordMcGregor-cbse5.pdf>

Stafford, J. & Wallnau, K. "Is Third Party Certification Necessary?" *Proceedings of the Fifth ICSE Workshop on Component-Based Software Engineering.* Toronto, Canada, May 2001, <http://www.sei.cmu.edu/pacc/CBSE4_papers/StaffordWallnau-CBSE4-22.pdf>

Stafford, J. & Wallnau, K. "Predicting Feature Interactions in Component-Based Systems." *Proceedings of the Workshop on Feature Interaction of Composed Systems* (held in conjunction with the 15th European Conference on Object-Oriented Programming). Budapest, Hungary, June 2001.

Wallnau, K. "Managing Design Complexity: Ensembles a Conceptual Language for Component-Based Design" (Invited Talk). ABB and Mälardalen University. Sweden, August 2001.

Wallnau, K. "Packaging Prediction with Prediction-Enabled Component Technology" (Invited Talks). NASA Ames Research Center and University of Texas, Austin. October 2001.

Wallnau, K. "Predictable Assembly from Certifiable Components: An Agenda for Applied Research" (Invited Talk). IFIP WG2.4. Cape Town, S. Africa, March 2002.

Wallnau, K. "Prediction-Enabled Component Technology" (Invited Talk). Advanced Real Time Systems Technology (ARTIST) ESPRIT Project. Paris, France, April 2002.

Wallnau, K. *Volume III: Technical Concepts of Predictable Assembly from Certifiable Components* (CMU/SEI-2002-TR-030). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University (in development).

Wallnau, K.; Stafford, J.; Hissam, S.; & Klein, M. "On the Relationship of Software Architecture to Software Component Technology." *Proceedings of the 6th International Workshop on Component-Oriented Programming* (WCOP6) (held in conjunction with the European Conference on Object Oriented Programming [ECOOP]). Budapest, Hungary, 2001.

8 Quality Software Development @ Internet Speed

8.1 Purpose

The explosion of electronic commerce on the Internet and the rapid rate at which corporations are reinventing themselves into e-businesses have created a radically new environment for software development. To be competitive in the new digital economy requires the ability to productively develop high-quality software systems at "Internet speed." However, even without the high-speed development demands of digital markets, the widespread diffusion of quality software development practices has met with serious obstacles. These are even more exacerbated in the Internet environment, with its emphasis on reduced cycle times and agility (the ability to deliver products quickly and to adapt to changing requirements quickly) [Aoyama 98].

The purpose of this research was to discover how quality and agility can be achieved in Internet software development. Specifically, the goals of this study were to (*a*) understand how and why the development of Internet software is different from traditional software development, (*b*) capture innovative practices used to achieve quality and agility in Internet software development and codify them at a high level, and (*c*) explore the situational factors that motivate the choice of these practices and the situations under which these practices are effective.

8.2 Background

Beyond the high-visibility Internet software leaders such as Microsoft and Netscape, very little is known about how Internet software product development is carried out in practice. How do software developers in this exploding market actually build their fast-cycle-time software? What is the impact of these software development practices on the quality of the software? Anecdotal evidence suggests that quality assurance practices are candidates for compression in cycle-time reduction [Wetherbe 00]. However, there is a need to understand how firms are developing fast-cycle-time software, what quality processes are used, and how software quality is retained in this rapid development environment.

Software development organizations are struggling to find the appropriate methods, tools, and practices that help to meet the challenges imposed by Internet-speed software development. Internet software development is characterized by rapid changes in requirements and unpredictable product complexity. Increased product development agility [Thomke 98; Iansiti 97] and methods that achieve a balance between flexibility and disciplined methodology [Cusumano 99a; Cusumano 99b] are necessary for survival in such an environment. However, most current software development methods and tools address the needs of large systems developed by multiple large teams and are not easily downward scalable to Internet-speed software development [Laitinen 00; Fayad 00; Toy 01].

Recognizing the need to address the concerns of high-speed software development, attempts have been made to tailor complex methodologies such as Rational Unified Process to Internet software development [Conallen 99]. There is growing interest in "light methodologies," such as spiral models of development, Agile Software Process (ASP), Scrum, and eXtreme Programming (XP). Evaluating XP from a CMM perspective, Paulk [Paulk 01] suggests that several ideas in XP contribute to achieving CMM level 2 and 3 practices.

While the current literature recognizes that many challenges and some new ideas are emerging, the need for solutions and frameworks that an Internet software development organization can readily adopt remains unfulfilled.

8.3 Approach

The research was conducted in three phases in a mixed-methods research design [Tashakkori 98] involving the collection of both qualitative and quantitative data. During phase 1, detailed case studies of Internet software development were conducted with nine companies in the Pittsburgh and Atlanta areas. The firms ranged in size from 20 to more than 100,000 employees and were in different industries in the private and public sectors. Some were Internet application start-ups, while others were established "brick and mortar" firms. The research team developed and tested a standard interview script and conducted semi-structured interviews in the fall of 2000 with senior managers, project managers, quality assurance personnel, and software developers in the firms. The interviews focused on eliciting the firms' development methods and tools, quality issues, product issues, development team and people-related issues. Interview data were supplemented with secondary data on the firms' competitive strategies and demographics. The team coded the resulting data and evaluated it using a concurrent mixed data analysis strategy involving grounded theory analysis and cross-case comparisons of qualitative data, including causal network analysis, cluster analysis, and cross-tabular analysis of quantitized data.

The objectives of phase 2 were to synthesize knowledge on promising practices for quality and agility in Internet software development and to develop an initial contingency matrix that

characterizes the situational factors motivating the choice of these practices and the situations under which they are effective. To facilitate achievement of those objectives, in October, 2001, the research team held a one-day Discovery Colloquium called "Quality Software Development @ Internet Speed." The activities in the colloquium were designed to (*a*) expand understanding of the interface between business issues and Internet software development through open dialogue, (*b*) discover and explore promising practices for Internet software development, and (*c*) engage participants in a vision-based approach to identify emerging and promising models, strategies, and directions to address current and downstream challenges in Internet software development. Other purposes of the Colloquium were to (*a*) share findings of the phase 1 research with the interviewed companies and (*b*) to check observations from phase 1 and determine whether other questions or issues should be addressed in further interviews in phase 3.

During phase 3, the validation of the contingency matrix and generation of ideas for further study continued through a second round of interviews with some of the original nine companies (those still in business) and with selected additional companies. Grounded theory analysis was conducted on the resulting data, building on the results from the previous phases. A cross-case comparison will be performed to characterize key differences in fast-paced development before and after the dot.com bust.

8.4 Collaborations

The principal collaborators in this study were

- Richard Baskerville, professor and chair, Department of Computer Information Systems, Georgia State University
- Jan Pries-Heje, associate professor, IT University of Copenhagen
- Linda Levine, senior member of the technical staff at the Software Engineering Institute
- Balasubramaniam Ramesh, associate professor, Department of Computer Information Systems, Georgia State University
- Sandra Slaughter, associate professor, Graduate School of Industrial Administration, Carnegie Mellon University

Will Hayes, another member of the technical staff at the SEI, contributed to the study during July through October, 2000. In addition, JoLee Loveland Link, of Volvox, Inc. (previously a visiting scientist at the SEI) and John Link, also of Volvox, Inc., contributed to the development of the Discovery Colloquium.

8.5 Evaluation Criteria

The research was expected to (*a*) produce findings that contribute to the practical outcomes of electronic commerce applications that are of better quality, have shorter time to market, and are cheaper, and (*b*) update understanding and set direction for next-generation improvement, including models for rapid technology development and deployment.

Work products included a report describing the Discovery Colloquium, interim and final briefings, and journal articles and conference presentations based on the results of the research. The Discovery Colloquium also served an informal test function, providing a loose validation of findings from phase 1.

8.6 Results

The findings from the study suggest that the practices used to develop software for the Internet differ sharply from traditional models of software engineering. For example, in the firms studied, development cycles are driven by features and are brief in duration, ranging from 15 days to 3 months. In the preliminary results from phase 1, the primary driver was always time. In order to be useful, competitive, or interesting, software products had to be rapidly brought to market. In that time-driven environment, quality was negotiable (it became a function of negotiations in the market between software competitors and customers), the development culture evolved (into less structure, smaller team sizes, diverse team compositions, and more emphasis on individuality), and processes adjusted.

A causal network aligning strategies, products, and processes developed by the team indicated that software processes are contingent on a firm's overall strategic context. A job shop (people-centric) approach can be used where the customer base is small and there are highly differentiated needs; however, if the customer base grows, the job shop approach may become too labor-intensive, costly, and inefficient. A more standard "batch" or process-centric approach appears to be appropriate when the product is less differentiated and the firm is serving a focused market segment; however, if customers demand more differentiation, this approach may not be the best suited, since the process is rather inflexible. Finally, the "assembly" or component-based approach appears well suited to a mass market strategy where there are sufficient common needs that standard components can be built and reused across products. (The assembly approach used by the firms in the study was more flexible than the traditional assembly-line production processes and seems closer to the current concept of "mass customization" in manufacturing.)

The researchers identified patterns of software process solutions that are appropriate for certain types of strategic contexts of Internet software development:

- parallel development—parallel releases; design, development, and quality assurance (QA) performed in parallel stages; coding ambiguous requirements
- frequent releases—smaller scope; feature slip; fluid requirements
- dependence on tools—integrated design and coding; tools doing most of the work
- customer implantation—the customer is a member of the team; requirements chunking and prioritization
- architecture focus—the architecture is standardized across applications, enabling parallel development
- component-based development—reuse; component interoperability
- minimal maintenance—phase 1 results showed only sketchy design or requirements documentation and short-lived products and versions; however, maintenance has become necessary as Internet firms have matured
- frequent tailoring of the methodology—the nature of the product is changeable; the composition of the development team is changeable; detailed method evolves day-to-day; just enough process to be effective; skip phases or tasks where necessary

The study thus provides evidence to support a contingency paradigm for software development by linking strategic context to software process choices. Many of the frameworks for software development implicitly assume that firms should adopt similar models and practices; in some cases tailoring guidelines are provided with the expectation that the model or practice can be translated to a particular context. The viewpoint resulting from this study differs from both the "one size fits all" approach and the "everything is unique and context-dependent" approach. From a pragmatic perspective, "one size fits all" may not be effective in software development, because not all development projects and environments are alike. Similarly, a completely context-dependent approach may be excessively costly to implement. The study results provide value by offering patterns of software process solutions that are appropriate for certain types of strategic contexts.

As the study progressed, factors affecting strategic context changed. Results from phase 3 analyses show the dot.com bust and tighter economy pressuring Internet software firms into placing less emphasis on speed and more on making business cases for product choices. Customers have come to expect speed and to require higher quality. Software firms are thus motivated both to "get the right product" and to "get the product right." Firms that survived the dot.com bust have gained experience, in conjunction with sets of technical solutions, and now have access to an ample supply of capable developers. These new factors are enabling firms to balance more of the requirements of quality software development at Internet speed and adjust to the challenges of a maturing technology in a contracting economy.

8.7 Publications and Presentations

8.7.1 Works in Progress

Ramesh, B.; Pries-Heje, J.; & Baskerville, R. "Internet Software Engineering: A Different Class of Processes." To appear in *Annals of Software Engineering*.

Baskerville, R.; Pries-Heje, J.; Ramesh, B.; & Slaughter, S. "Six Silver Bullets: Costs and Benefits of High Speed Software Development Practices." Submitted to *Communications of the ACM* in August 2002 and accepted for review.

Baskerville, R.; Levine, L; Pries-Heje, J.; Ramesh, B.; & Slaughter, S. "A Framework for Software Development in Parallel Economies: Industrial and Knowledge-Based Software Organizations." Submitted to *Information Technology and Management* special issue on business application of advanced software engineering, September 2002.

Slaughter, S.; Levine, L.; Baskerville, R.; Pries-Heje, J.; & Ramesh, B. "Linking Software Processes to Competitive Strategies: A Product-Process Matrix for Internet Application Development." Submitted in August 2002 to *Management Information Systems (MIS) Quarterly* and in review.

Comparative analysis of findings from phases one and three is currently being worked. Once this analysis is complete and the results documented, a final article will be written and submitted for publication.

Short papers may also be written on Internet speed with respect to agility and continuous innovation.

8.7.2 Published/Presented Works

Baskerville, R.; Levine, L.; Pries-Heje, J.; Ramesh, B.; & Slaughter, S. *Discovery Colloquium: Quality Software Development @ Internet Speed* (CMU/SEI-2002-TR-020). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. <http://www.sei.cmu.edu/publications/documents/02.reports/02tr020.html>.

Baskerville, R.; Levine, L.; Pries-Heje, J.; Ramesh, B.; & Slaughter, S. "Balancing Quality and Agility in Internet Speed Software Development." International Conference on Information Systems, Dec 15–18, 2002, Barcelona, Spain. <http://dsi.esade.es/icis2002/>.

Baskerville, R.; Levine, L.; Pries-Heje, J.; Ramesh, B.; & Slaughter, S. "How Internet Software Companies Negotiate Quality." *Computer 34*, 5 (May 2001): 51–57.

Baskerville, R.; Levine, L.; Pries-Heje, J.; Ramesh, B.; & Slaughter, S. "Divergent Practices for Speed and Agility in Internet Software Development" (panel). 22nd Anniversary International Conference on Information Systems, New Orleans, Louisiana, Dec. 17–20, 2001. Association for Information Systems, 2001.

Pries-Heje, J.; Levine L.; & Slaughter, S. "Engineering Software @ Internet Speed." *Proceedings of the 11th International Conference on Software Quality.* Pittsburgh, Pennsylvania, Oct. 22–24, 2001. Milwaukee, WI: American Society for Quality, 2001.

Baskerville, R. & Pries-Heje, J. "Racing the E-Bomb: How the Internet is Redefining Information Systems Development Methodology." *Realigning Research and Practice in IS Development: The Social and Organisational Perspective*. Edited by B. Fitzgerald, N. Russo, & J. DeGross. New York: Kluwer, 2001, 49–68.

Baskerville, R. & Pries-Heje, J. "Emethodology: Towards a Systems Development Methodology for E-Business and E-Commerce Applications." *Developing a Dynamic, Integrative, Multi-Disciplinary Research Agenda in E-Commerce/E-Business*. Edited by S. Elliot, K. V. Andersen, P. Swatman, & S. Reich. Ourimbah, New South Wales: BICE Press, 2001, 164–178.

Baskerville, R. & Pries-Heje, J. "Information Systems Development @ Internet Speed: A New Paradigm in the Making!" 10th European Conference on Information Systems, Gdansk, Poland, June 6–8, 2002.

Baskerville, R.; Levine, L.; Pries-Heje, J.; Ramesh, B.; & Slaughter, S. "Divergent Practices for Speed and Agility in Internet Software Development" (panel). 22nd Anniversary International Conference on Information Systems, New Orleans, Louisiana, December 18, 2001.

Baskerville, R. & Ramesh, B. Presentation on Internet Speed research approach and findings. Seminar, Department of Computer Information Systems, Georgia State University, Atlanta, Georgia, November 2, 2001.

Baskerville, R. "Emethodology: Towards a Systems Development Methodology for E-Business and E-Commerce Applications." Kilpisjarvi Information Systems Seminar, Finland, November 10, 2001.

Pries-Heje, J. Buzztalk at the IT University of Copenhagen. Copenhagen, Denmark, November, 2001.

Pries-Heje, J. Presentation of research results at two Danish companies, PKA and VALTECH. Copenhagen, Denmark, November, 2001.

Baskerville, R.; Ramesh, B.; Levine, L.; Pries-Heje, J.; & Slaughter, S. "Quality Software Development @ Internet Speed: Findings and Discussion." Discovery Colloquium, Software Engineering Institute, Pittsburgh, Pennsylvania, October 18, 2001.

Baskerville, R. "Emethodology: Towards a Systems Development Methodology for E-Business and E-Commerce Applications." Colloquium, Temple University, Philadelphia, Pennsylvania, October 16, 2001.

Baskerville, R., & Pries-Heje, J. "Racing the E-Bomb: How the Internet Is Redefining Information Systems Development Methodology." WG 8.2 Working Conference, Boise, Idaho, July 28, 2001.

Baskerville, R. & Pries-Heje, J. "Emethodology: Towards a Systems Development Methodology for E-Business and E-Commerce Applications." TC8 Working Conference, Salzburg, June 23, 2001.

Slaughter, S. "Internet Software Development Practices." Presented at a meeting for visiting Japanese software executives, Graduate School of Industrial Administration, Carnegie Mellon University, May 17, 2001.

Levine, L. "The Role of Process/Process Improvement in Internet Speed Development" (panel). Presented at a meeting for visiting Japanese software executives, Graduate School of Industrial Administration, Carnegie Mellon University, May 17, 2001.

Levine, L. & Slaughter, S. "Quality Software Development @ Internet Speed: Interim Findings, Future Directions." Software Engineering Institute, Pittsburgh, Pennsylvania, May 14, 2001.

Slaughter, S. "Developing Software at Internet Speed." Presentation to Software Industry Center (SWIC) members (presentation secured financial support from SWIC for 2001–2002), Heinz School, Carnegie Mellon University, March 22, 2001.

Levine, L. & Pries-Heje, J. "Process Improvement @ Internet Speed." Software Engineering Process Group National Meeting, New Orleans, Louisiana, March 13, 2001.

9 Learning from Software Development and Acquisition Failures

9.1 Purpose

The Learning from Software Development and Acquisition Failure IR&D project was initiated to investigate how the SEI could make a strategic difference in preventing or reducing software-intensive system failure. The project was particularly focused on the question of why software projects and programs fail and what to do about these failures. The goal was to formulate a recommendation outlining how the SEI could develop a strategic response to this serious and ongoing problem.

Currently available software engineering technologies are only a partial solution to the problem. While failure continues to be proclaimed at the global level of public discourse in the media, software project difficulties and failings are not sufficiently acknowledged or understood at the local or organizational level. Software engineering methods and techniques will only become solutions in organizations that are honestly engaged in investigating their strengths and weaknesses and that are committed to act on their findings. Organizations need to become inquiring systems that learn from past failures to appreciate what can be gained from the discipline of software engineering. Even more important, organizations need to learn how to apply this discipline in particular contexts.

The goal of this IR&D project has been to determine how the SEI, with its leading edge knowledge of software engineering techniques and its regular exposure to troubled and failing projects through ITAs and other engagements, can build on its current position and assume a leadership role in moving from assessment to transition of software engineering discipline into practice. While the SEI has, in the past, sensitized organizations to their software engineering problems, it can also play an important role in providing a problem-solving and knowledge-management scaffolding that will facilitate this transition. The infrastructure that the SEI could provide would enable organizations to continually apply viable technical solutions to their software engineering problems in particular contexts.

9.2 Background

Published literature over the last 20 years has been reviewed. The most important findings are discussed in this section.

9.2.1 The Dimensions of Failure

Surveys on software project outcomes provide information on the frequency of software failure. The Standish Group has published several influential surveys of software project failure [Standish 95, Johnson 99].

Table 1: Standish Project Failure Surveys

	Project Resolution Rate		
	Successful[11]	Challenged	Failed
1994	16%	53%	31%
1996	27%	33%	40%
1998	26%	46%	28%
2000	28%	49%	28%

KPMG reports a similar failure rate from a survey in 1994, where 62% of respondents reported at least one project had exceeded budget by at least 30% during the last five years [KPMG 95]. While Table 1 shows a slight improvement in project success rates over time, which the Standish Group attributes to smaller project size, better project management, and more use of standard infrastructures [Johnson 99], the failure rates are unacceptably high. On the other hand, Barry Boehm has pointed out that equating project termination with project failure (as is done in Table 1) can put the wrong slant on what it means to be mismanaged and ill conceived.

Reviewing the Standish reports, Boehm examines project termination causes and points out that in many cases well-conceived and well-managed projects nevertheless should be terminated for the same reasons that poorly managed projects are [Boehm 00]. He makes the point that relative to a failure rate of 40% for new product introductions and the 50% average failure rate for sales leads, a 31% termination rate might be regarded as less than a crisis level. In Boehm's view, stigmatizing project termination as "failure," with negative consequences to careers and reputations, is less desirable than acknowledging project termination as an expected and legitimate management option for a proportion of software projects. Some very well managed projects need to be terminated because they are found not to fit new corporate

[11] The Standish Group classifies projects into three resolution types:
- Successful: The project is completed on time and on budget, with all features and functions as originally specified.
- Challenged: The project is completed and operational, but over budget, over the time estimated, and with fewer features and functions than initially specified.
- Failed: The project is cancelled before completion [Johnson 99].

strategies or meet changes in customer needs. Boehm's insights are reinforced by work on the de-escalation of commitment.

De-escalation of Commitment

Once commitment to a project is established, even a failing project can prove resistant to change (and therefore a difficult target for failure mitigation): "[T]hese projects seem to take on a life of their own, continuing to absorb valuable resources, while failing to deliver any real business value. While prior research has shown that managers can easily become locked into a cycle of escalating commitment to a failing course of action, there has been comparatively little research on de-escalation, or the process of breaking such a cycle" [Montealegre 00, p. 417]. Montealegre and Keil define a model of de-escalation in four steps: (1) problem recognition, (2) reexamination of prior course of action, (3) search for alternative courses of action, and (4) implementing an exit strategy. Kiel and Robey developed a model for project de-escalation: "Central to our model is the notion that 'bad news' on a project must be communicated from the actor(s) who are in a position to observe it to the actor(s) who are in a position to do something about it. This means overcoming the 'mum effect,' a reluctance to transmit bad news. But transmittal alone is not always enough, for in many cases the bad news is communicated to those who are unwilling to listen, a phenomenon we call the 'deaf effect.' De-escalation requires overcoming both the deaf effect and the mum effect" [Kiel & Robey 99, p. 83].

9.3 Approach

The traditional literature review was supplemented by a computational analysis of online information sources relevant to software failure, including an analysis of online articles collected from key software engineering journals and information from specialized repositories on risk and DoD-specific problems. The computational analysis also demonstrated how records of ongoing practice can be used to create knowledge that typically remains hidden from either reflective practitioners or researchers.

9.3.1 Analysis of Online Sources

The analysis of online information sources confirms the published literature review in showing that software project failure is still far too prevalent. However, just as important, the analysis showed that, by and large, the research community has somewhat less interest in the failure problem than practitioners, especially military practitioners. The online analysis provided a glimpse of how different communities, academic, practitioner, and military, treated the topic of software-intensive system failure. Interesting differences were found in the range and types of topics in published journals as against online sources. Published literature in software engineering journals focuses on failures in system performance, whereas online sources such as Peter Neumann's *Risk Digest* and military databases (Defense Technical Information Center and Center for Army Lessons Learned (CALL)) expanded the scope of dis-

cussion considerably. In fact, the latter contained discussions of causes of software failure as traceable to inadequate communication between developers and acquirers, whereas little mention of this was made in the academic research journals culled.

Although much more work needs to be done in the area of capturing and analyzing software engineering practice, analysis of online repositories suggests that important practical information can be extracted from the experiences of projects and programs and used as a basis for solving practical problems. Capturing what transpires in software engineering practice can become an important basis for establishing software engineering discipline, and the SEI is in a position to lead the transition. However, the SEI's current intervention approaches, such as Independent Technical Assessments (ITAs), Team Risk Evaluations (TREs), and CMM assessments, need to be supplemented in various ways to realize their discipline-establishing potential. The SEI needs to create an infrastructure that supports longer-term engagements that maintain longer-term links to practitioners. For example, revisiting organizations that have been sensitized to their software-intensive system development or acquisition problems will increase the probability that software engineering discipline will emerge (see Section 3 and Appendix C of *Learning from Software Project Failure* for a fuller account of analysis methods).

9.3.2 Field Work

Three fieldwork engagements were done. Two involved recording retrospective experiences from practitioners and one consisted in participation on an Independent Technical Assessment (ITA). The retrospectives consisted of intense interviews of key players or stakeholders in relatively large, troubled projects. The first retrospective was a case study of a university financial management system. The second was a shorter-term study of the development of a complex software engineering model, especially with reference to the deployment of information technology to support its development. Both retrospectives were undertaken to gain experience with the techniques and tools available to conduct field engagements that incorporated practical goals of improving software engineering practice. The aim of participation in the ITA was to evaluate the knowledge management support of the SEI's ITA process. It was found that ITAs have great potential in not only bringing back very useful information on the nature of software-intensive system project or program failure in organizations that are an important part of the SEI's target audience but in making a strategic impact on the transition of software engineering discipline. Currently this potential is not being fully realized. An important goal of this IR&D project has been to formulate how ITAs and other SEI field engagements might be extended to better realize this potential.

The Emphasis on Stories and Action Research

One of our research participants provided an excellent example of how collecting stories in the context of process improvement for systems and software environments would be useful:

There is one in every group... they ask, "What's the ROI for all this process engineering stuff?" In my office, I have a filing cabinet with four drawers full of ROI data pointing to the effectiveness of process improvement for engineering... "How do you want it?" I then ask, offering several categories of breakdown by project, and to which the typical reply is, "Well, besides that what?"

Our interviewee went on to say that the most effective *data*—data that influences people's actions and real change—is the first-hand story, primarily derived from the first brave individuals who have come back to other engineers like themselves to report that their lives have been improved, that projects are more easily managed and are more predictable in the playing out of the life cycle, more calculable in scope, and so on. Action research is a methodology for producing such actionable stories that can be derived from field engagements.

Action research involves practitioners in the act of information gathering for understanding the real contexts of work for the purposes of learning and improvement. This may or may not always involve a clear relationship to more exacting numerical analysis methods. The whole purpose of this approach is to capture anecdotes, analyze them, understand them, and intervene in a disciplined, measured manner. Action research can be integrated into the fabric of everyday work in a way that can provide meaningful information without the overhead of more intrusive and potentially quite artificial strictures of statistical research design.

9.4 Collaborations

Ira Monarch, senior member of the SEI technical staff, was the project lead. Peter Capell, SEI visiting scientist, and Neal Altman, member of the SEI technical staff, were primary investigators. Their collaborations took the following forms:

- collaborated with 10-person team from the SEI on the preparation of and outputs from a three day on-site ITA supported by the assessed organization
- interviewed 25 people from the Carnegie Mellon University community who were both developers and users of a system being developed at Carnegie Mellon. The interviews lasted an hour or more. Each of the participants donated their time.
- interviewed 20 people who had participated in the creation of a software engineering model. The group consisted of both members of the SEI and members of external organizations. The interviews lasted an hour or more. All interviewees donated their time.

There were also shorter interactions with the following people, who all donated their time: Dolores Wallace, NASA; Michael Koo, NIST; Charles Bush, Defense Contract Management West, Defense Logistics Agency; Larry Fry, Siemans Medical Systems; Darren Dalcher, Forensic Systems Research Group, School of Computing, South Bank University, London; Bill Everett, SPRE (consulting group); Wade Shaw, Avionics failure; Larry Bernstein, Senior Industry Professor, Computer Science, Stevens Institute of Technology; Mary Glaser, CALL

Repository, Army; Ted Marz, SEI data from ITAs; Dave Zubrow, OSD Tri-Service Assessment Initiative; Chinnarao Mokkapate, Company Problem Reports and Reviews (NDA with USS); Jeff Drezner, Rand Corporation; Randy Weinberg & Larry Heimann, Information Systems Program, Carnegie Mellon.

9.5 Results and Criteria of Evaluation

The problems of IT deployments, currently very widespread, very much in the news, and often apparently very painful to those involved, have not yielded to the current suite of process improvement or technology transition templates and methodologies available in the software engineering marketplace. These problems are emerging from a common source of missing discipline in adopting software engineering and project management best practices. The templates come in many forms, some of which assist in implementing team processes, or TQM, or software process improvement. Models are a form of template that are created in order to "show the way," in theory, to isolate the critical variables from the vast list of variables that are possible in the programmatic and organizational milieu.

A cottage industry has grown up around templates that are dedicated to tracking changes to the templates and interpreting and translating them for host organizations. While this symbiosis is successful on one level, the now-growing membership of experiences in and stories of very large-scale information technology failures suggests the need for additional approaches. These problems require locally generated solutions. Specific solutions can be adapted from more general solutions but only with significant local commitment and problem solving. These solutions will not come through further statistical analyses of data or through more and more top ten lists of best and worst practices, but through collaborative networks of problem solvers who are learning from one another in and across projects and programs. The results of this IR&D project not only point in this collaborative networking direction but also provide a proposal for how the SEI can leverage its current respected position to become a hub in this network and lead the transition of software engineering discipline into practice.

9.5.1 Learning from Failures

Independent Technical Assessments (ITAs) and other forms of SEI assessment and evaluation create a space and a time inside the organizations assessed where honest accounts of problems have a chance to be aired and addressed. Unfortunately the SEI has not, up to now, adequately supported this problem-identification and problem-solving mode of intervention. The key point is that software engineering methods and techniques will only be perceived and function like solutions in an organization that is honestly engaged in investigating its strengths and weaknesses and has the commitment of all its members to act on the results of the investigation. In other words, one way in which patterns of software project failure can be turned around in organizations is for organizations to become more adept at capturing stories

of failure and success and addressing these stories through collaborative problem solving and continued knowledge networking.

Stories are important because organizational inquiry is not scientific inquiry. People are engaged by stories and can readily identify with the predicaments described. Metrics are fine and can serve a useful purpose, but applying the right metrics in a given context depends on prior communication, interaction, and identification of key problems and potential solutions.

Collaboration is important because all relevant voices need to be heard and taken into account. Stories are an important vehicle for getting all the voices heard and the different perspectives considered. However, group processes and infrastructures are also needed to support the extended dialog it takes to formulate, test, and refine various hypotheses that are responsive to all the relevant stories and that will converge to a solution in a timely fashion.

Knowledge networks internal to the organization are needed so that stories describing predicaments can be shared sufficiently to enable the identification of problems that need to be solved. External knowledge networks are needed because potential solutions are often found by exploring outside the organization.

While there is still quite a bit to learn about the nature of sharing stories, collaborative problem solving, and knowledge networking, enough is known to see that, taken together in a systematic whole, they offer a viable approach to addressing the current patterns of software project failure. Such inquiring systems, if deployed in many organizations, provide a viable approach for the transition of software engineering knowledge, technology, and discipline into wide practice.

9.5.2 Spawning Inquiring Systems in Organizations

The SEI is already engaged in work that, suitably enhanced, could help its client organizations take an inquiring system approach to recurring patterns of software project failure. We include here a sketch of a proposal to explore and prototype what the SEI will have to do to augment its already existing processes and resources in order to assume a leadership role in this approach to software project failure and the transition of software engineering discipline into practice. We propose augmenting the SEI's current ITA process and other SEI field engagement approaches so they have the potential to spawn inquiring systems inside the SEI's target organizations.

Suitably augmented ITAs will help the SEI's client organizations take an inquiring system approach to software project failure. However, software project and program failure is not necessarily simply a problem within single organizations. Software engineering discipline and the technologies that support it gain acceptance through co-evolution of multiple inter-

connected artifacts and the network of people who come to be connected to them and to each other. This co-evolution takes place through the slow creation and evolution of networks of people and supporting technologies that interact locally. The approach to augment ITAs and other forms of action research at the SEI proposes to support local problem solving through more global knowledge networking.

Currently ITAs consist of an information-gathering phase and a briefing phase that rolls up findings and makes recommendations. The augmentation would add the following:

1. In the information gathering stage
    a. provide techniques, templates, and examples enabling context to be adequately described, so those who did not participate can make sense of the information
    b. gather observations from different assessees and assessors and organize them according to whether they contradict, confirm, or elaborate one another
2. Organize and cross-link information gathered, including context, observations, findings, and recommendations, and have them accessible from a Web site that includes
    a. observations and context linked to all findings listed in the briefing, with findings linked to recommendations
    b. competing hypotheses in the form of causal narratives that apply in and across customer projects based on gathered observations, contexts, and outcomes
    c. common themes across ITAs
3. Track assessed projects over extended periods and evaluate the impact of ITA findings and recommendations by
    a. maintaining contact between the SEI and assessed organization participants
    b. updating hypotheses and themes according to the outcomes collected
    c. summarizing information into common themes and outcomes across organizations

In addition to personnel executing the current ITA process, a separate knowledge integrator will be needed to perform these information- and knowledge-intensive tasks. A major reason for augmenting the ITA process is to increase understanding of and impact on

- how organizations are actually facing well-known problems, how they are actually trying to address them, and whether and how they are using software engineering technologies
- the effectiveness of the SEI's interventions, including whether any new awareness or new behaviors emerge and what outcomes these have
- the diagnostic and problem-solving capabilities of organizations, including how these may be improved
- how long-term contact with the SEI can make a difference, including introduction and extended use of various SEI software engineering technologies
- fostering the creation of innovation and improvement networks linking producers of software-intensive systems to acquirers, and linking these to the SEI (see publications

and presentations below, especially *Learning from Software Project Failure,* for a fuller account)

9.6 Publications and Presentations

Monarch, Ira. "Learning from Software Project Failure and TNKM." The New Knowledge Management: A Generational Shift, sponsored by: Knowledge Management Consortium International (KMCI), George Washington University, and Homeland Defense Journal, March 20, 2002, Arlington, Virginia. <http://www.marketaccess.org/event_km_20mar.asp>.

Monarch, Ira; Capell, Peter; & Altman, Neal. "Learning from Cases: Beyond Success and Failure," prepared for presentation at the cancelled 2001 SEI Symposium.

Monarch, Ira. "Understanding Software Engineering Failure as Part of the SWEBOK." *Proceedings of the 14th Conference on Software Engineering Education and Training*. Charlotte, North Carolina, Feb. 19–21, 2001. New York: IEEE Computer Society Press, 2001.

Monarch, Ira; Capell, Peter; & Altman, Neal. *Learning form Software Project Failure*. SEI technical report draft (available from the authors). January 2002.

10 Technology Change Management in High-Maturity Organizations

10.1 Purpose

The purpose of this study was to

- examine the practice of technology change management (TCM) to gather the best thinking and examples of effective applications, with an eye toward making that content available to the SEI community—both internal and external
- consider the technical and business case for further work in TCM at the SEI

In particular, several high-maturity or high-performing organizations appear to be garnering competitive advantage by way of a strategic focus on TCM. Prior to the IR&D study, anecdotal reports of advantages included increased business capture, speedier process improvement, and attainment of higher maturity levels as measured against the Capability Maturity Model® for Software (SW-CMM), successful uptake of new technologies that affect organizations' ability to innovate and deliver high-quality products, and consequent improvements in time to market and market share.

What are the practices of these organizations? Can they be codified? Are they adequately reflected in the relevant practices of CMMI®? Can we use what we learn to make lower-maturity organizations more effective as well? Are technology developers at the SEI poised to gain advantage from effective practice in TCM as modeled by high-performing organizations? Can we learn practices from industry and transition them to DoD organizations? The purpose of this study was to consider these questions.

10.2 Background

The SEI had never performed a concerted study of the state of the practice in TCM. Teams of researchers and practitioners at the SEI did, of course, investigate transition and adoption models, build on their own experience, perform scores of interventions and customer applications, share what they knew, and embody their knowledge of technology transition in various

® Capability Maturity Model and CMMI are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

outputs, such as presentations, courses, and consulting. However, the SEI had not deliberately captured this knowledge as an organizational asset or recommended practice for use by others. While individual researchers continue to study and learn, too little effort has been made to capture current, effective practice of TCM, codify it, make it routinely available at the SEI, use it as input for products, or share it with our customer and collaborator community.

In the last few years, a great deal of thinking and work has been done in TCM outside the SEI, although little of it is framed systematically. Numerous books and other documents, Web sites, thinkers, and consultants feature aspects of technology adoption or change management as part or all of their content and offerings. For example, Stanford University has a graduate program on managing change, Lake Superior College has an undergraduate course on managing change, practitioners have diagnostic tools to assess an organization's readiness to manage technical change (some of them available free online), researchers publish a best practices report on change management, professional societies offer certificates and membership programs for TCM, and a few researchers are even attempting to write algorithms on the diffusion and adoption of innovations. However, nearly all of the research, tools, and training is oriented to the adoption of a single technology (or the implementation of a single reengineering or organizational change), and not a repeatable, persistent, systematic practice.

Also, while the results of this exploratory IR&D study will not fill the gap for a state of the practice report, it will set a baseline, indicate credible continuing research possibilities, and get benchmarks for further work.

10.3 Approach

We briefly surveyed the available literature to orient ourselves to the current body of knowledge on TCM. Our emphasis was on practice rather than theory, and our purpose was to gather relevant models of and thinking on effective TCM practice.

We performed case studies of three high-performing (mostly high-maturity) organizations with TCM practices that we had reason to believe were effective and producing the desired organizational results. We chose these organizations from a set of candidates that were willing to work with us in this capacity. We intended the number of case studies to be small, only three or four qualitative cases from which to set a baseline. To select the cases, we first looked at organizations that reported themselves at level 5, given that TCM is a key practice at that level. We performed a search for organizations that had published or presented in some capacity on how technology change management is pursued in their organization. Also, we decided to keep the cases in large, defense-oriented organizations for this initial cut—follow-on research could look at the TCM practices in other industries, smaller organizations, and the like.

To focus our research method, we looked at several sources to guide our case-study technique and hypothesis generation. We found that a GAO report on case study techniques was particularly helpful, and designed our case studies following some of their advice for several of their case study types.[12] In addition, we looked at qualitative research techniques, and settled on an inductive technique to generate our hypothesis, to be tested with qualitative field experience, not unlike the "grounded theory" method. Put simply, grounded theory calls for the investigators to pose an explicit theory, observe field conditions to test the theory, revise the theory based on that experience, observe again with the revised theory, and so forth. It is a qualitative technique driven by data.

Based on our literature review, we attempted to craft a model of TCM practice, but concluded there was too little information to do so based on the literature. We did attempt to code the findings from literature but decided we were enforcing patterns where few were yet to be found. We found abundant literature on technology change, change management, and the like, but little of it lends itself to frameworks or describes consistent, systematic, repeated practice across an organization. In the absence of a clear reference model, we devised a looser set of topic areas of interest, and formulated a set of questions to guide the case studies based on literature and our own experience.

We identified two potential risks to our approach, but found in practice that they did not become a problem. First, because an SEI team would be on-site with the studied organizations getting detailed information about a practice that is also a key practice in the SW-CMM, we were concerned that the study would be confused with CMM-based assessments. We were careful to communicate that these studies were not assessments, and found that organizations did not have a problem with this. In fact, we were probably helped by the experience of these organizations in having assessments, because they were accustomed to openly sharing information about their practices and communicated effectively about them.

The second risk we identified was the possible danger that the studied organizations would not care to have the results published, and this would make organizations reluctant to participate. Again, this was not an issue. We mitigated it first by giving the studied organizations the option to control how results were published. The options were not to publish at all, to publish with all organization references sanitized, or to publish with the organization clearly identified. The organizations have so far been eager to pursue full publishing, and appear to consider it valuable to have an SEI case study performed.

One risk we did not identify up front affected us throughout the study. The vigorous trend of merger and acquisition and reorganization in this market sector caused us to lose two of the organizations we originally pursued for case studies, and also affected two of those we did

[12] United States General Accounting Office, Case Study Evaluations, Maryland, GAO/PEMD-91-10.1.9, November 1990, pp. 9–10.

study. As both of the organizations that had to decline because of mergers and reorganizations had published or provided provocative information on their TCM practices, we found this quite disappointing. In particular, one of the organizations had been pursuing TCM since the very beginning of their process improvement efforts, and has communicated in workshop settings that it helped them to achieve higher levels of maturity sooner and adopt technologies that even other high-maturity organizations have found too challenging. It is unfortunate that we could not perform a case study on this organization.

Finally, we did notice one accidental side benefit as we developed our sources and approach. Two other SEI teams that we know of have profited from the case study know-how that we developed, because we informally diffused this knowledge to them.

10.4 Collaborations

Three case studies are complete as of this writing and in revision for publishing. A potential fourth case is currently being negotiated but is unlikely to be available in the remaining IR&D schedule. Three cases are large defense contractors pursuing process improvement against the CMM or CMMI. Two of the points of contact, TCM leads in their organizations, have continued to collaborate and seek further collaboration with the SEI on these topics because of their experience in the IR&D study.

Eileen Forrester, a member of the Accelerating Software Technology Adoption initiative (ASTA), leads this study. We included members of other SEI units (COTS-Based Systems, Product Line Practice, and Software Engineering Process Management) as part of the study team, both to get the benefit of their expertise and to foster natural information flows for the study results in other programs. We included a CMM author to get the perspective intended by the CMM developers when they wrote TCM content for the model. One original outside team member, with long experience in both software and change management, has since joined the SEI. All other team members were drawn from ASTA, because of the focus of that team on technology change and transition. All team participants have either a practice or research background in the study area.

A community of active researchers and practitioners convened in the summer of 1999 at a workshop on TCM sponsored by the SEI. This group recommended a list of next steps to be pursued, and this study takes up some of those steps. Workshop participants such as Stan Rifkin and John Vu have agreed to review our final outputs. Stan Rifkin was particularly helpful in pointing us to sources on qualitative research and case studies.

In addition, Barry Boehm and Larry McKee, who have been active in a community trying to foster the uptake of spiral development and evolutionary acquisition, have agreed to review and comment on our final report. We also intend to provide courtesy copies to the DoD

Change Management Center, the DoD Office of Technology Transition, and the Software Collaborators Network.

We have a large set of transition partners and SEI alumni, including past resident affiliates, who are likely to be interested in the output of this study. These are anticipated collaborators for follow-on activities, if any are undertaken.

We may consider offering existing communities of practice (such as Society of Learning, Knowledge Management Consortium, Innovation Network, etc.) and educational institutions that have an interest in TCM an opportunity to give us input or review of results once the study is complete.

Finally, study participants and working group participants from various conferences in which ASTA participates have expressed strong interest in participating in any post-study work to codify best TCM practices.

10.5 Evaluation Criteria

These are the criteria to evaluate the study results:

- Have we developed substantive information on the proposed key elements of effective TCM practice in high-performing organizations?
- Do the perceived business results of the studied organizations justify further investment in TCM?
  - Should the SEI embody this content in our offerings?
  - Do we need more data?
- Can the key elements of effective TCM practice be adopted in other organizations or are they unique to the specific organizations in which they are found? In particular, are the elements applicable to DoD organizations?
- Have we identified new or improved content on TCM for inclusion in CMMI?
- Will the elements be useful to SEI technical programs as they plan the adoption of their technologies?

10.6 Results

Here are the results to date against the success criteria.

1. These are the high-level results from examining the key elements of effective TCM practice in high-performing organizations. Greater detail can be found in the forthcoming case-study reports and technical note.

- Each of the three studied organizations defined a different scope for TCM. One organization employed TCM strictly for technologies that support their own development processes. A second used that definition as a subset, but used TCM primarily for technologies that would be included in products and services for customers. The third organization used TCM to guide deployment of newly defined or improved development processes. Results for these organizations varied with the definition in predictable ways. For example, TCM used in external products led to better business capture, TCM used to support internal development processes enabled faster process improvement and sustainment of high maturity, and TCM used to support process deployment led to more effective uptake of the particular processes.
- We saw a clear variation in practices and results between the two highest maturity organizations on the one hand, and the third case, which was a high performer but lower maturity. The third case in effect confirmed the findings of importance in the high-maturity cases.
- With that variation in mind, here are some of the features of the most effective TCM practice we have observed across the cases. These items are not exhaustive, but meant to be indicative of practices in organizations getting strong benefits from a focus on TCM.
- TCM has official program status, multiple senior roles participate, and the program and participants enjoy direct and regular access to the senior sponsor.
- The organization has a governing strategic technology planning process, of which their TCM program is a part. The TCM leaders are strong owners and participants—sometimes drivers—in this process and the organization knows the structure, schedule, outputs, and decision-making process for the technology plan.
- TCM decisions are tied to business results and therefore treated as serious business projects.
- Significant funds are provided for all TCM process and activities.
- Additional resources beyond funds are dedicated to TCM. These include hardware, training, and participation and capabilities provided by opinion leaders, technical gurus, enabling operational functions, and so forth.
- Sponsorship for TCM is authentic and visible, and cascades from the top and throughout management.
- TCM has active and continuing senior management visibility. TCM projects and decisions are monitored and reported on consistently at all levels.
- The TCM program or process has some kind of central decision making—there is a clear owner for TCM and the leader or leaders are senior, well-respected people.
- In addition to the clear owner and central decision making, there is wide sharing of responsibility for success, and broad participation is expected from all employees. TCM objectives are often included in performance objectives.
- The top performers have an explicit process for scanning, monitoring, and selecting new technologies (selecting is especially careful); these include exposure to external sources and participation with outside organizations as needed.
- A strong and repeatable piloting process (plus training) is always used once a technology is selected.

- Whole product and deployment are shaken down to a few elements proven to work in a specific organization's environment. That is, the highest performers with the most careful selection process need far less support to roll out a technology.

2. The business results of the studied organizations are overwhelmingly positive. They include support of process improvement, achievement of higher maturity levels with the attendant benefits, much improved rates of business capture, customer satisfaction, and profound, positive effects on strategic technology planning and innovation. The highest maturity organizations assert that even if acquired or merged, the organization-wide belief in the benefits of TCM would lead them to continue to pursue the same path, even if they had to do so "underground." Further, the high-maturity organizations assert that even lower-maturity organizations, those that never want more than level 2, and possibly those not pursuing formal process improvement at all could benefit by approaching TCM as they do.

   a. Given these results, it seems prudent to at least codify the TCM practices we find at level 5 and make these available to other process-aware organizations for further validation.

   b. We may be over-generalizing from too little data. Certainly our selection process was biased in favor of those already successful with TCM. We may need to seek some specific failure data to test the limits and the optimistic assertions of the successful high-maturity organizations.

3. The key elements of effective TCM practice are not conceptually difficult, but may require appropriate cultures and organizational styles most often found in process-oriented cultures. The elements may also be applicable to DoD organizations; some elements seen in the high-maturity cases seem particularly suitable to DoD style, with their phased approach, disciplined selection, and attention to maturity of technologies.

4. CMM and CMMI authors have already reacted in part to our findings and adjusted some content in the CMMI. They have also made an SEPG Conference presentation to clarify the definition of TCM—something we communicated through a team member who was also an author of the CMM and the presentation.

5. If SEI technical programs (DoD ATD, ACTD, and DARPA programs could also benefit) were aware of the monitoring and selection processes of major industry organizations practicing systematic TCM, they would be well-positioned to have their technologies considered for adoption by high performers. If adopted, these organizations are excellent early adopters and pilot sites for technologies and could provide superb references to later adopters.

10.7 Publications and Presentations

The chief output will be a report on the literature survey and case studies. If we decide the results are sufficiently compelling, we will also consider a strategy for dissemination of the information and infusion into SEI products for external and internal consumption. For example, we will recommend how promising practices may be incorporated into products for internal use, such as TransPlant, and for external use, such as Introducing New Software Technology and Managing Technological Change, and will give a series of presentations on the study results. We have submitted presentation proposals to two conferences to share the results to date and case-study participants will co-present.

11 Summary

All of the IR&D projects from the 2002 fiscal year have yielded valuable results that are being disseminated to the community through technical reports, journal articles, and presentations. Among the reports that have been or will be published describing the projects and their results in detail are the *Technical Roadmap for Enterprise Integration* for the Analysis of Enterprise Integration Applications project, *Flow-Service-Quality (FSQ) Engineering: Foundations for Network System Development* (CMU/SEI-2002-TN-019), *Perspectives on Open Source Software* (CMU/SEI-2001-TR-019), and *Discovery Colloquium: Quality Software Development @ Internet Speed* (CMU/SEI-2002-TR-020).

One of the projects, Predictable Assembly from Certifiable Components, has been selected for ongoing work as an SEI initiative. Further results from study in this area will be published on the SEI Web site.

References

[Aoyama 98] Aoyama, M. "Web-Based Agile Software Development." *IEEE Software 15*, 6 (November/December 1998): 56–65.

[Arief et al. 01] Arief, B.; Gacek, C.; & Lawrie, T. *The Many Meanings of Open Source* (CS-TR-737). Newcastle, England: Department of Computer Science, University of Newcastle upon Tyne, August, 2001.

[Bass 98] Bass, L.; Clements, P.; & Kazman, R. *Software Architecture in Practice*. Reading: Mass.: Addison-Wesley, 1998.

[Bass 01] Bass, Len; Buhman, Charles; Comella-Dorda, Santiago; Long, Fred; Robert, John; Seacord, Robert; & Wallnau, Kurt. *Volume I: Market Assessment of Component-Based Software Engineering Assessments* (CMU/SEI-2001-TN-007, ADA 395250). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu/publications/documents/01.reports/01tn007.html>.

[Boehm 00] Boehm, Barry. "Project Termination Doesn't Equal Project Failure." *Computer 33*, 9 (September 2000): 94–96.

[Britton 01] Britton, Chris. *IT Architectures and Middleware*. New York, NY: Addison-Wesley, 2001.

[CIO 01] CIO Council. *A Practical Guide to Federal Enterprise Architectures,* Version 1.1. February 2001.

[Conallen 99] Conallen, J. *Building Web Applications with UML*. Boston: Addison-Wesley, 1999.

[Crynkovic 01] Crynkovic, I.; Schmidt, H.; Stafford, J; & Wallnau, K. (Eds.). *Proceedings of the 4th ICSE Workshop on Component-Based Software Engineering: Component Certification and System Prediction,* <http://www.sei.cmu.edu/pacc/CBSE4-Proceedings.html>.

[Crynkovic 02a] Crynkovic, I.; Schmidt, H.; Stafford, J.; & Wallnau, K. (Eds.). *Proceedings of the 5th ICSE Workshop on Component-Based Software Engineering: Component Certification and System Prediction.* <http://www.sei.cmu.edu/pacc/CBSE5/CBSE5-Proceedings.html>.

[Crynkovic 02b] Crynkovic, I.; Schmidt, H.; Stafford, J.; & Wallnau, K. (Eds.). *Anatomy of a Research Project in Predictable Assembly* (White Paper). <http://www.sei.cmu.edu/pacc/CBSE5 /CBSE5_whitepaper.pdf>.

[Cusumano 99a] Cusumano, M. A. & Yoffie, D. B. "Software Development on Internet Time." *IEEE Computer 32*, 10 (October 1999): 60–69.

[Cusumano 99b] Cusumano, M. A. & Yoffie, D. B. "What Netscape Learned From Cross-Platform Software Development." *Communications of the ACM 42*, 10 (October 1999): 72–78.

[Fayad 00] Fayad, M. E.; Laitinen, M.; & Ward, R. P. "Software Engineering in the Small." *Communications of the ACM 43*, 3 (March 2000): 115–118.

[Feller et al. 02] Feller, J. & Fitzgerald, B. *Understanding Open Source Software Development* (ISBN: 0201734966). Boston, MA: Addison-Wesley Professional, UK, March 2002.

[Finkelstein 99] Finkelstein, Clive & Aiken, Peter. *Building Corporate Portals using XML*, New York, NY: McGraw-Hill, 1999.

[Hevner 01] Hevner, A.; Linger, R.; Sobel, A.; & Walton, G. "Specifying Large-Scale, Adaptive Systems with Flow-Service-Quality (FSQ) Objects," 110–120. *Proceedings of the 10th OOPSLA Workshop on Behavioral Semantics*. Tampa, Florida, Oct. 14–18, 2001. Boston: Northeastern University, 2001.

[Hevner 02a] Hevner, A.; Linger, R.; Sobel, A.; & Walton, G. "The Flow-Service-Quality Framework: Unified Engineering for Large-Scale, Adaptive Systems." *Proceedings of the 35th Annual Hawaii International Conference on System Science (HICSS35).* Hawaii, Jan. 7–10, 2001. Los Alamitos, CA: IEEE Computer Society Press, 2002.

[Hevner 02b] Hevner, A.; Linger, R.; Pleszkoch, M.; A.; & Walton, G. "Flow-Service-Quality (FSQ) Engineering for Specification of Complex Systems." *Practical Foundations of Business and System Specifications*, Edited by H. Kilov and K. Baclawski. New York: Kluwer, Inc., 2002.

[Hissam 02] Hissam, S.; Moreno, G.; Stafford, J.; & Wallnau, K. "Packaging Predictable Assembly." *Proceedings of the 1st IFIP/ACM International Working Conference on Component Deployment.* Berlin, Germany, June, 2002. <http://www.sei.cmu.edu/staff/kcw/cd2002.pdf>.

[Iansiti 97] Iansiti, M. & MacCormack, A. "Developing Products on Internet Time." *Harvard Business Review 75*, 5 (September-October 1997): 108–117.

[Johnson 99] Johnson, Jim. "Turning CHAOS into SUCCESS." *Software 9*, 3 (12/99–01/00): 30-34, 39. <http://www.softwaremag.com/archive/1999dec/Success.html> (1999).

[JV 00] *Joint Vision 2020.* U.S. Government Printing Office, June 2000.

[Keil and Robey 99] Keil, Mark & Robey, Daniel. "Turning Around Troubled Software Projects: An Exploratory Study of the Deescalation of Commitment to Failing Courses of Action." *Journal of Management Information Systems 15*, 4 (Spring 1999): 63–88.

[KPMG 95] KPMG. "Runaway Projects—Cause and Effects." *Software World* (UK) *26*, 3 (1995): 3–5.

[Laitinen 00] Laitinen, M.; Fayad, M. E.; & Ward, R. P. "The Problem With Scalability." *Communications of the ACM 43*, 9 (September 2000): 105–107.

[Larsson 02] Larsson, M.; Wall, A.; Norström, C.; & Crynkovic, I. "Using Prediction Enabled Technologies for Embedded Product Line Architectures." *Proceedings of the 5th ICSE Workshop on Component-Based Software Engineering.* Orlando, Florida, May 2002. <http://www.sei.cmu.edu/pacc/CBSE5/Larsson-cbse5-final.pdf>.

[Leonard 00] Leonard, A. Salon Free Software Project: Chapter 1, "Boot Time" [online]. <http://www.salon.com/tech/fsp/2000/03/06 /chapter_one_part_2/index.html> (October 2001).

[Li 02] Li, P.L.; Shaw, M.; Stolarick, K.; & Wallnau, K. "The potential for synergy between certification and insurance." *Proceedings of the First International Workshop on Software Reuse Economics* (held in conjunction with the Seventh International Conference on Software Reuse). Austin, Texas, April 16, 2002. <http://www.sei.cmu.edu/staff/kcw/icsr02.pdf>.

[Linger 02a] Linger, R.; Pleszkoch, M.; Walton, G.; & Hevner, A. *Flow-Service-Quality (FSQ) Engineering: Foundations for Network System Development* (CMU/SEI-2002-TN-019). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. <http://www.sei.cmu.edu/publications/documents/02.reports /02tn019.html>.

[Linger 02b] Linger, R.; Pleszkoch, M.; Walton, G.; & Hevner, A. "Flow-Service-Quality Requirements Engineering For High Assurance Systems Development." *Proceedings of the International Workshop on Requirements for High Assurance Systems*, Essen, Germany, Sept. 9, 2002. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002.

[Magee 97] Magee, J.; Kramer; & Giannakopoulou, D. "Analysing the Behaviour of Distributed Software Architectures: A Case Study," 240-247. *Proceedings of the Fifth IEEE Workshop on Future Trends of Distributed Computing Systems.* October 1997.

[Mead 00] Mead, N.; Ellison, R.; Linger, R.; Longstaff, T.; & McHugh, J. *Survivable Network Analysis Method* (CMU/SEI-2000-TR-013, ADA383771). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. <http://www.sei.cmu.edu/publications/documents/00.reports/00tr013/00tr013title.html>.

[Meyer 00] Meyer, B. "Contracts for Components." *Software Development Online Magazine* <http://www.sdmagazine.com/documents/s=742/sdm0007k/0007k.htm>.

[Montealegre 00] Montealegre, Ramiro & Keil, Mark. "De-escalating Information Technology Projects: Lessons from the Denver International Airport." *MIS Quarterly 24*, 3 (September 2000): 417–448.

[Moore 01] Moore, A.; Ellison, R.; & Linger R. *Attack Modeling for Information Security and Survivability* (CMU/SEI-2001-TN-001). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu/publications/documents/01.reports/01tn001.html>.

[Moreno 02] Moreno, G.; Hissam, S.; & Wallnau, K. "Statistical Models for Empirical Component Properties and Assembly-Level Property Predictions: Toward Standard Labeling." *Proceedings of the 5th ICSE Workshop on Component-Based Software Engineering*. Orlando, Florida, May 2002. <http://www.sei.cmu.edu/pacc/CBSE5/Moreno-cbse5-final.pdf>.

[Necula 96] Necula, G. C. & Lee, P. "Safe Kernel Extensions Without Run-Time Checking." *Proceedings of the Second Symposium on Operating Systems Design and Implementation (OSDI '96).* Seattle, Washington, October 28-31, 1996.

[Noble 97] Noble, Brian D.; Satyanarayanan, M.; Narayanan, Dushyanth; Tilton, J. Eric; Flinn, Jason; & Walker, Kevin R. "Agile Application-Aware Adaptation for Mobility." *Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP).* Saint-Malo, France, October 1997. New York, NY: Association for Computing Machinery, 1997.

[OSI 01a] Open Source Initiative. "The Open Source Definition Version 1.8" [online]. <http://www.opensource.org/docs/definition.html> (2001).

[Paulk 01] Paulk, M. "Extreme Programming from a CMM Perspective." *IEEE Software 18*, 6 (November/December 2001): 19–26.

[PITAC 00] President's Information Technology Advisory Committee (PITAC), Panel on Open Source Software for High End Computing: co-chairs Smarr, L. & Graham, S. "Developing Open Source Software to Advance High End Computing" [online]. <http://www.ccic.gov /pubs/pitac/pres-oss-11sep00.pdf> (September 11, 2000).

[Prowell 99] Prowell, S.; Trammell, C.; Linger, R.; & Poore, J. *Cleanroom Software Engineering: Technology and Practice.* Reading, MA: Addison Wesley, 1999.

[Public Law 96] Public Law 104-106, Section 5125, 110 Stat. 694 (1996).

[Raymond 99] Raymond, E. *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary* (ISBN: 1565927249). Cambridge, MA: O'Reilly & Associates, October 1999.

[Stafford 01] Stafford, J. & Wallnau, K. "Is Third Party Certification Necessary?" *Proceedings of the 5th ICSE Workshop on Component-Based Software Engineering.* Toronto, Canada, May 2001. <http://www.sei.cmu.edu/pacc/CBSE4_papers /StaffordWallnau-CBSE4-22.pdf>

[Stafford 02] Stafford, J. & McGregor, J. "Issues in Predicting the Reliability of Composed Components." *Proceedings of the 5th ICSE Workshop on Component-Based Software Engineering.* Orlando, Florida, May 2002. <http://www.sei.cmu.edu/pacc/CBSE5 /StaffordMcGregor-cbse5.pdf>.

[Standish 95] The Standish Group. *The CHAOS Report (1994).* <http://www.pm2go.com/sample_research/chaos_1994_1.asp> (1995).

[Tashakkori 98] Tashakkori, A. & Teddlie, C. "Mixed Methodology: Combining Qualitative and Quantitative Approaches." *Applied Social Research Methods 46*, 9 (September 1998): 171–179.

[Thomke 98] Thomke, S. & Reinertsen, D. "Agile Product Development: Managing Development Flexibility in Uncertain Environments." *California Management Review 41*, 1 (Fall 1998): 8-30.

[Toy 01] Toy, B. "E-business from the Front Line of a 140-Year-Old Brick and Mortar Company." *Transportation Journal 40*, 4 (Summer 2001): 27–33.

[UL 98] Underwriter Laboratories. *UL-1998, UL Standard for Safety for Software in Programmable Components.* Northbrook, IL, 1998.

[Wetherbe 00] Wetherbe, J. & Frolick, M. "Cycle Time Reduction: Concepts and Case Studies." *Communications of AIS 3* Article 13, 1–42.

[Zachman 97] Zachman, John A. "Enterprise Architecture: The Issue of the Century." *Database Programming and Design* (March 1997).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	October 2002	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
SEI Independent Research and Development Projects	F19628-00-C-0003

6. AUTHOR(S)

Steve Cross, Eileen Forrester, Scott Hissam, Rick Kazman, Linda Levine, Rick Linger, Tom Longstaff, Ira Monarch, Dennis Smith, Kurt Wallnau

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2002-TR-023

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	ESC-TR-2002-023

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

Each year, the Software Engineering Institute (SEI) undertakes several Independent Research and Development (IR&D) projects. These projects serve to (*a*) support feasibility studies investigating whether further work by the SEI would be of potential benefit and (*b*)support further exploratory work to determine if there is sufficient value in eventually funding the feasibility study work as an SEI initiative. Projects are chosen based on their potential to mature and/or transition software engineering practices, develop information that will help in deciding whether further work is worth funding, and set new directions for SEI work. This report describes the IR&D projects that were conducted during fiscal year 2002 (October 2001 through September 2002).

14. SUBJECT TERMS	15. NUMBER OF PAGES
agent-based architecture, mobile devices, enterprise integration, Flow-Service-Quality Engineering, network design, data fusion, network intrusions, open source software, software components, predictable assembly, component-based development, Internet-speed software development, software-intensive system failure, software project failure, high-maturity organizations, technology change management	100

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL